Intra-Procedural Analysis

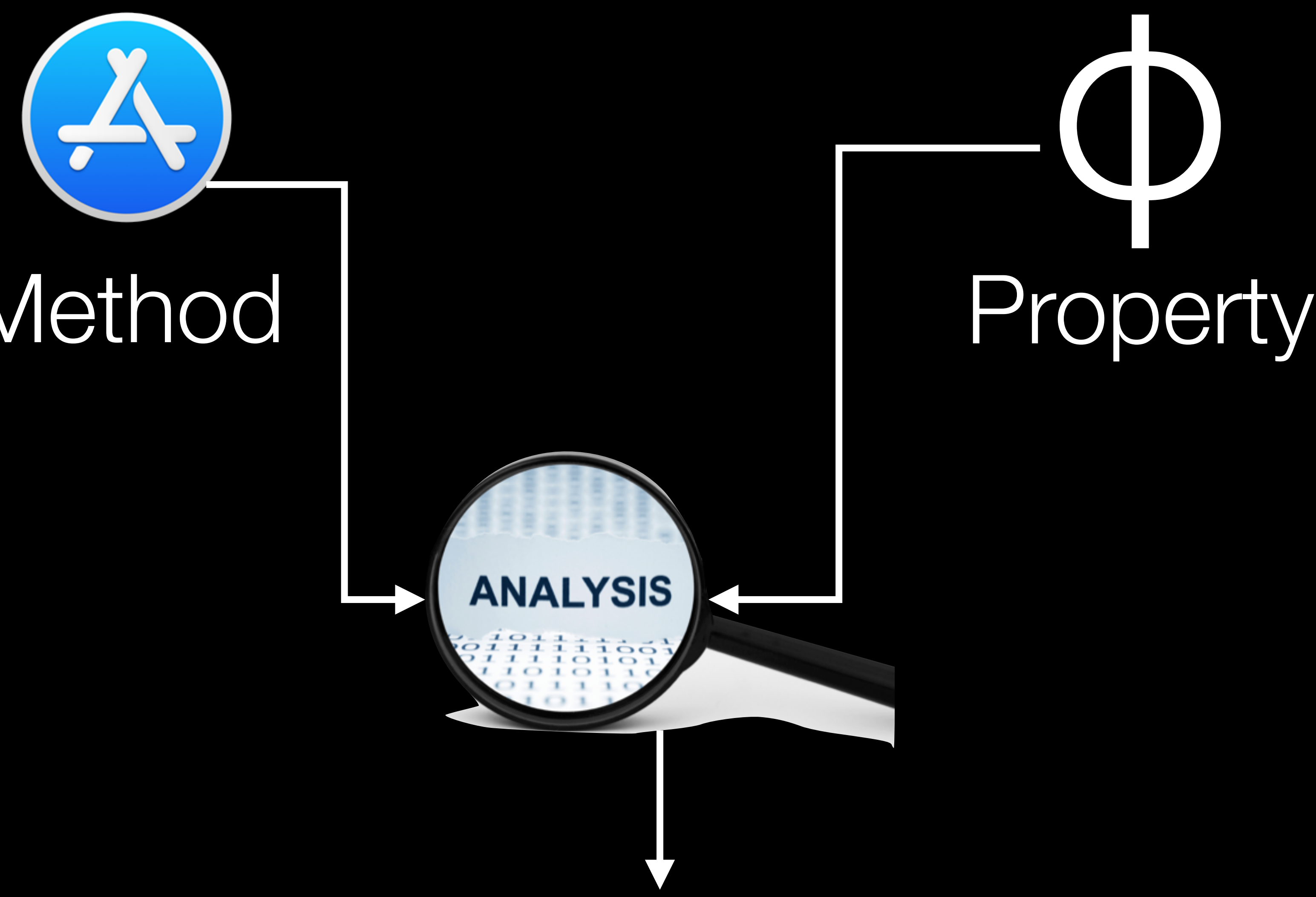
CMPUT 416/500
Foundations of Program Analysis

Karim Ali
@karimhamdanali

Previously

- Static analysis is undecidable
- Sample analyses
- Intermediate representations
- Case study: Java and Android

Intra-Procedural Analysis



Does the property hold at statement S?

Property	Analysis
Is this variable still used later on?	Live-Variables Analysis
Can this code ever execute?	Dead-Code Analysis
Can this pointer ever be null?	Nullness Analysis
Is this file handle ever closed?	Typestate Analysis

Let's consider this code

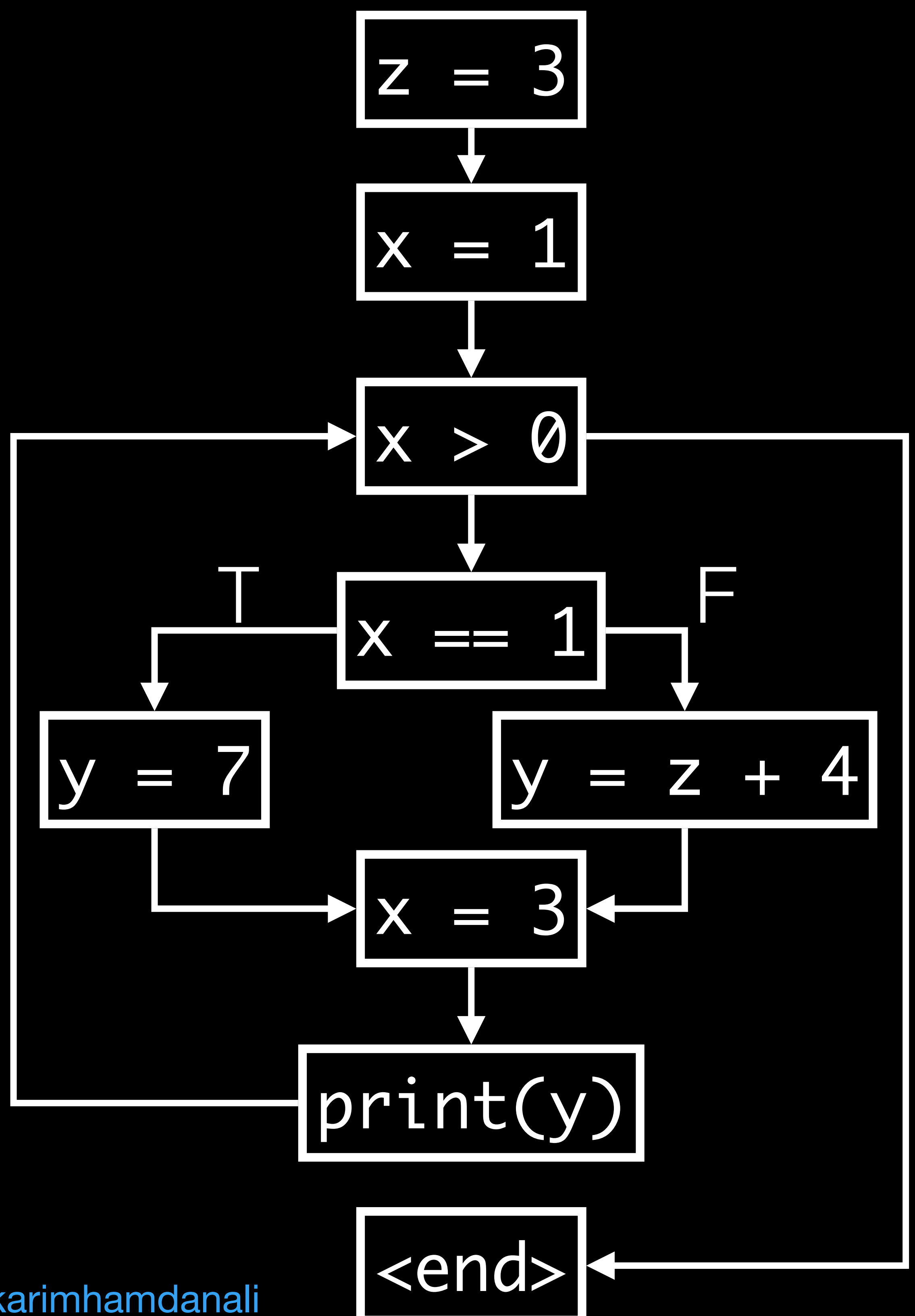
```
z = 3;
x = 1;
while(x > 0) {
  if(x == 1)
    y = 7;
  else
    y = z + 4;
  x = 3;
  print(y);
}
```

Let's consider this code

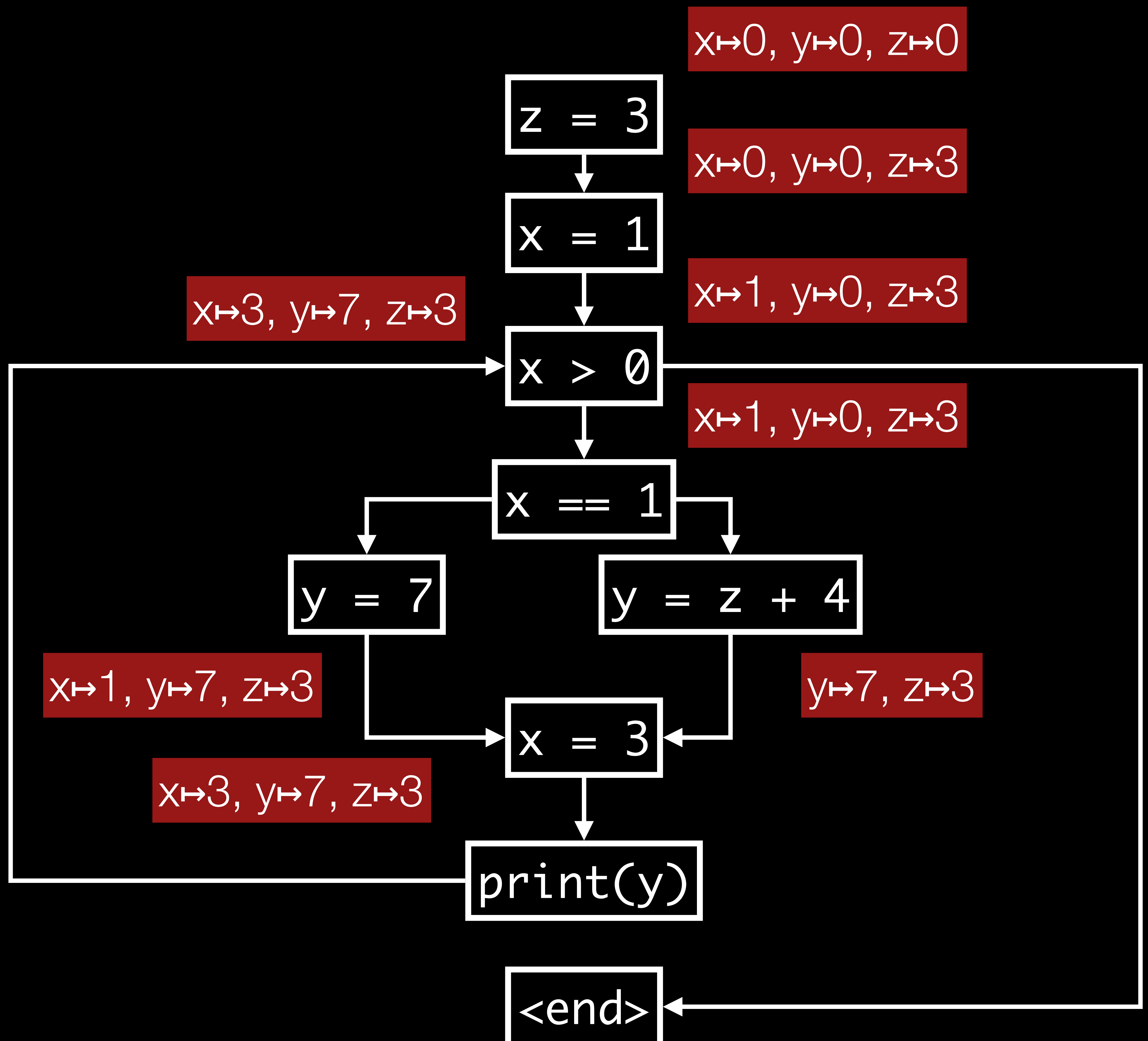
- Which variables carry constant values?
- Which values do they exactly carry?

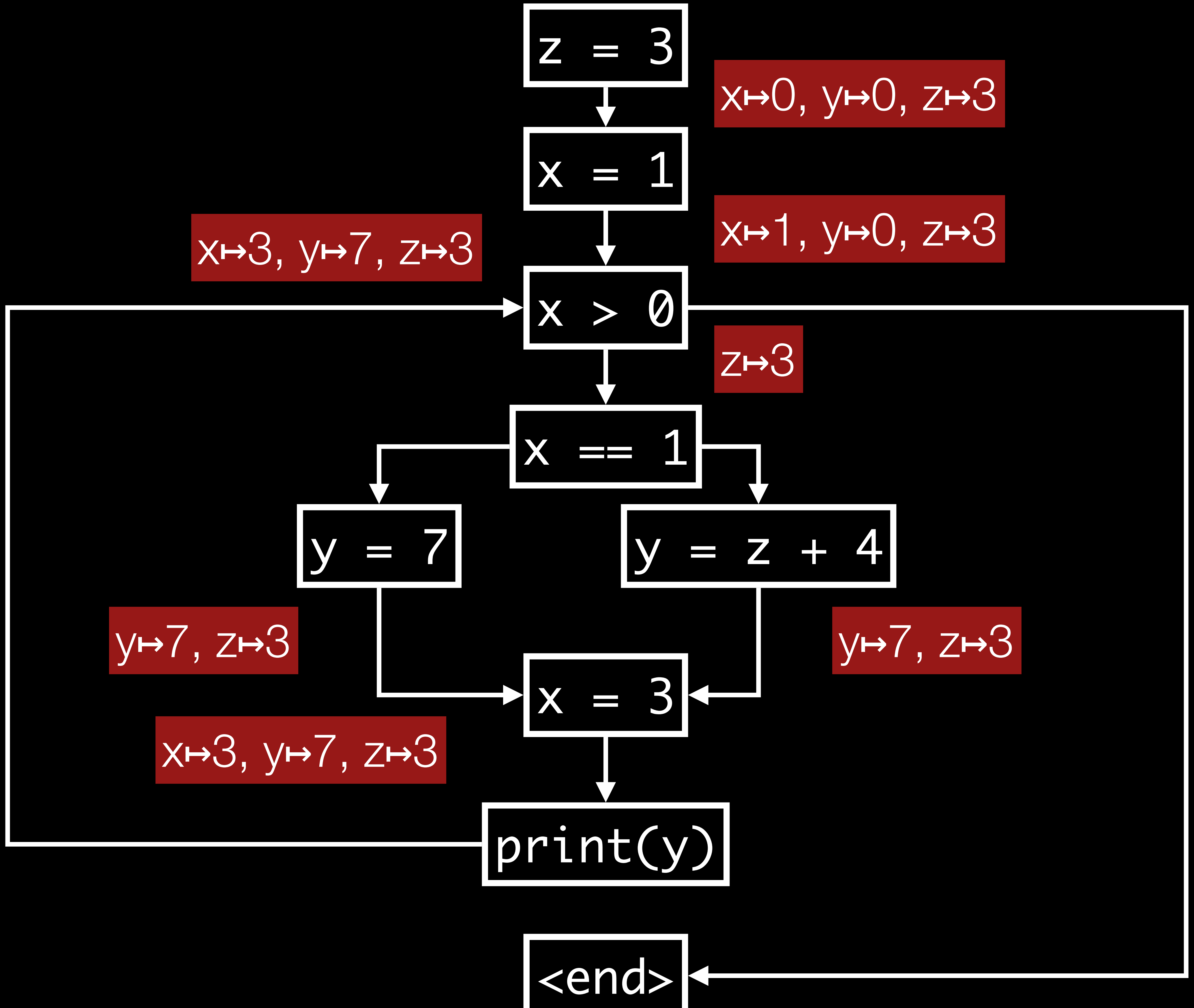
```
z = 3;
x = 1;
while(x > 0) {
  if(x == 1)
    y = 7;
  else
    y = z + 4;
  x = 3;
  print(y);
}
```

Control-Flow Graph




```
z = 3;
x = 1;
while(x > 0) {
  if(x == 1)
    y = 7;
  else
    y = z + 4;
  x = 3;
  print(y);
}
```

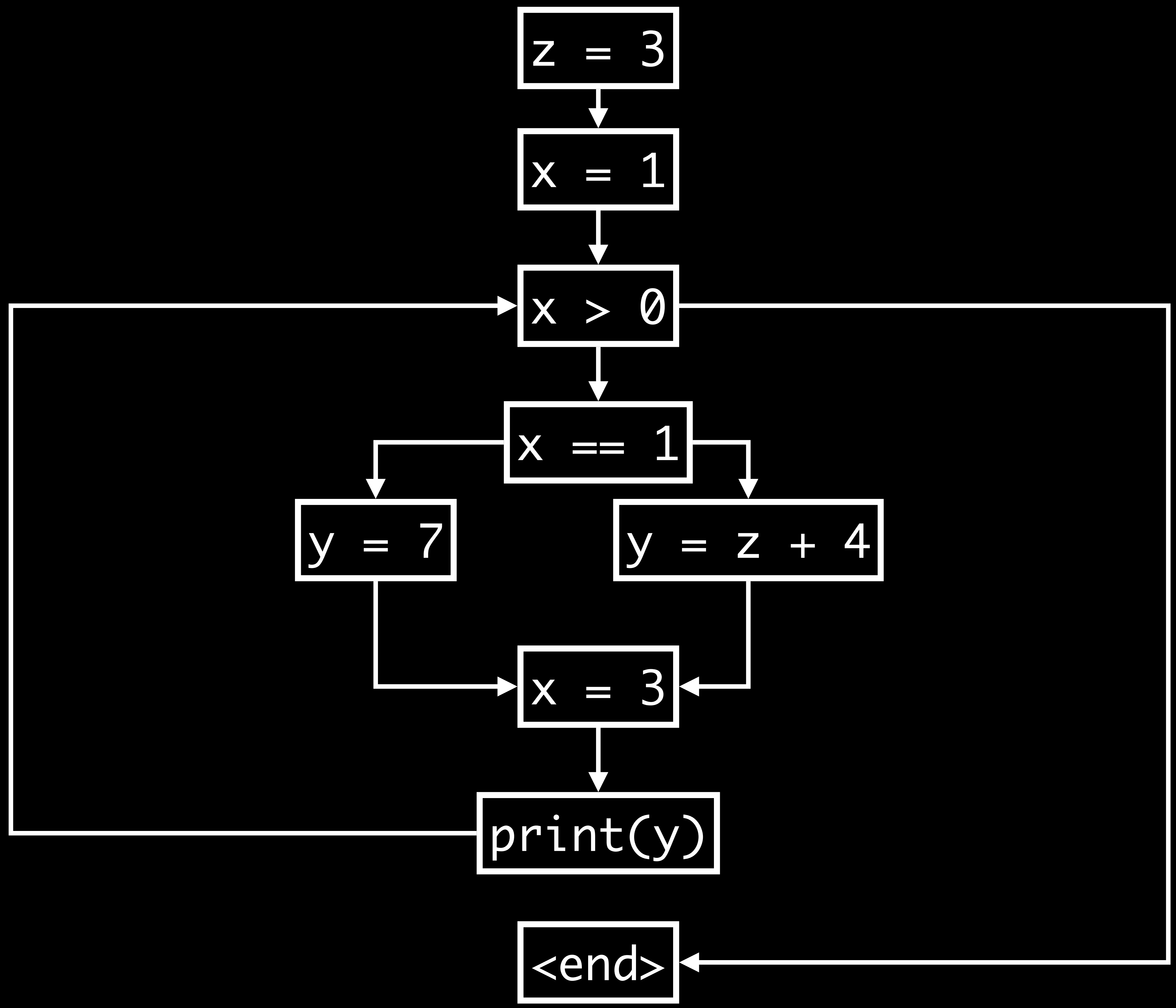

x↦0, y↦0, z↦0
z = 3
x↦0, y↦0, z↦3
x = 1
x↦1, y↦0, z↦3
x↦3, y↦7, z↦3
x > 0
x↦1, y↦0, z↦3
x == 1
y = 7
y = z + 4
x↦1, y↦7, z↦3
y↦7, z↦3
x = 3
x↦3, y↦7, z↦3
print(y)
<end>

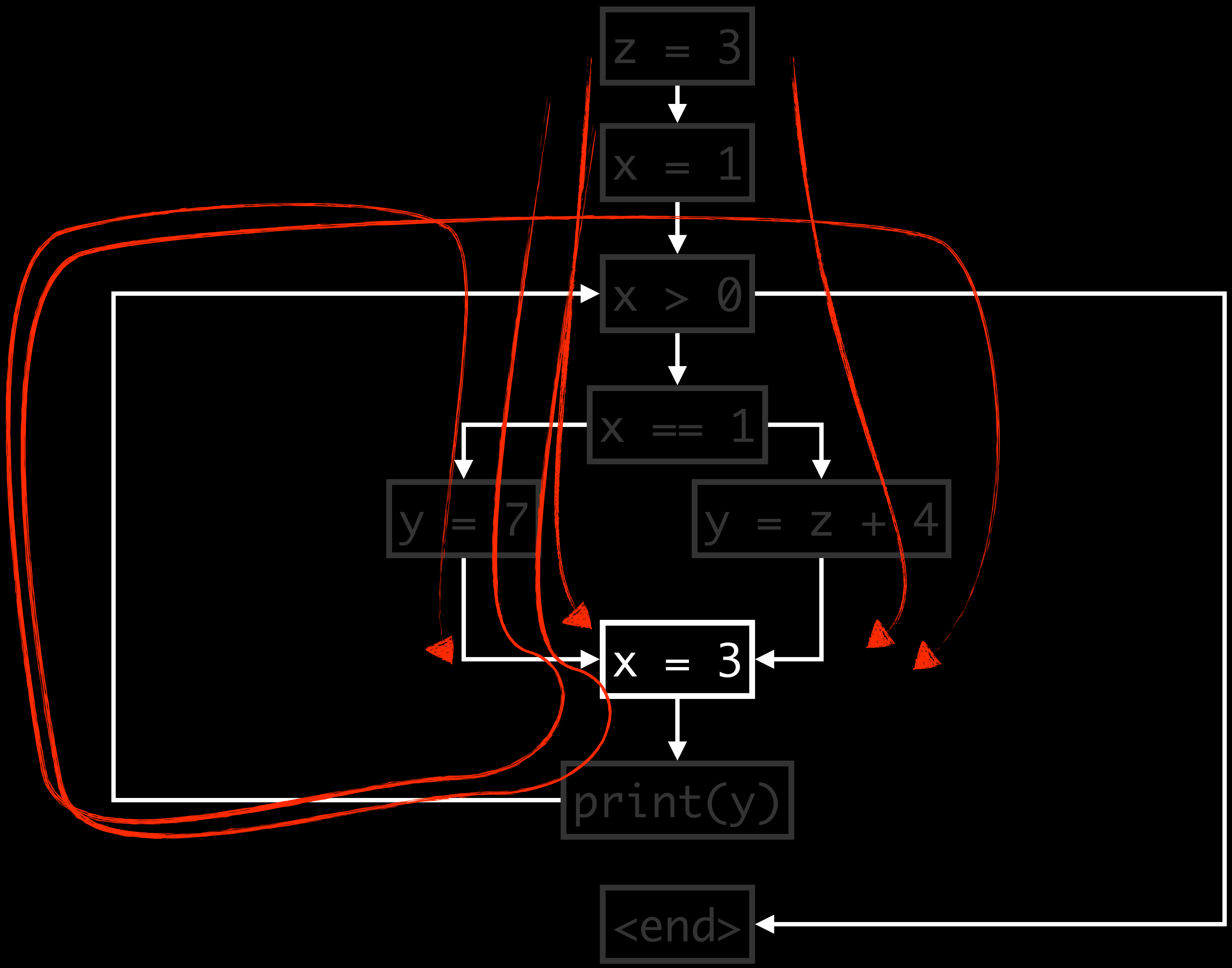

z = 3
x↦0, y↦0, z↦3
x = 1
x↦1, y↦0, z↦3
x↦3, y↦7, z↦3
x > 0
z↦3
x == 1
y = 7
y = z + 4
y↦7, z↦3
y↦7, z↦3
x = 3
x↦3, y↦7, z↦3
print(y)
<end>

🤔

… how can we find a general solution?

Naive “Solution” is Meet Over All Paths


z = 3
x = 1
x > 0
x == 1
y = 7
y = z + 4
x = 3
print(y)
<end>


z = 3
x = 1
x > 0
x == 1
y = 7
y = z + 4
x = 3
print(y)
<end>

… how do we compute Meet Over All Paths Solution?

MOP Solution

initial value

$$\forall s \in Stmt : MOP(s) = \sqcup \{f_p(i) \mid p \text{ is a path from } s_0 \text{ to } s\}$$

composed “flow function” for path

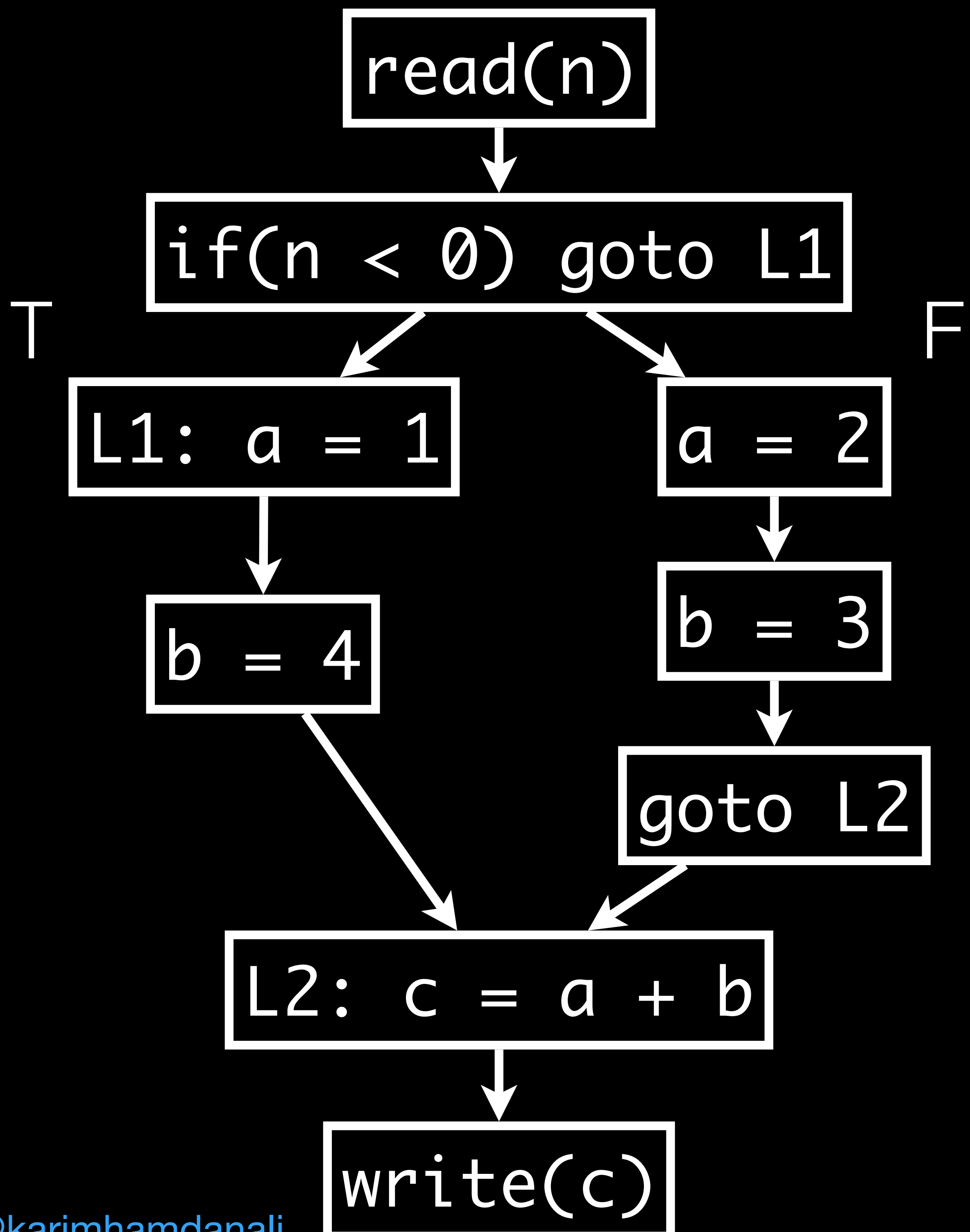
Post Correspondence Problem

Generally Uncomputable
[Kam, Ullman 1977]

Let’s consider this code

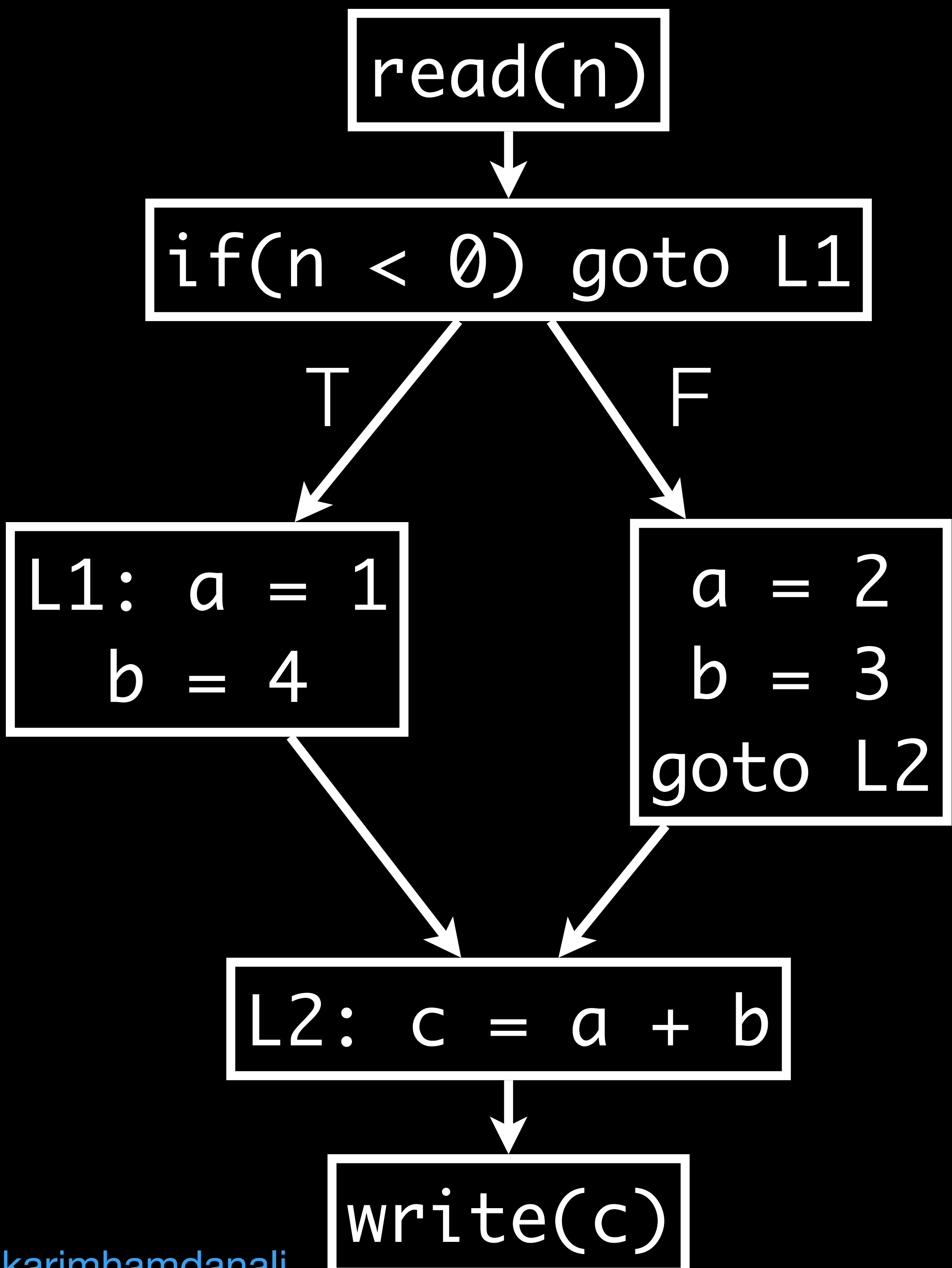
```
read(n);

if(n < 0) {
  a = 1;
  b = 4;
} else {
  a = 2;
  b = 3;
}

c = a + b;
write(c);
```

Control-Flow Graph



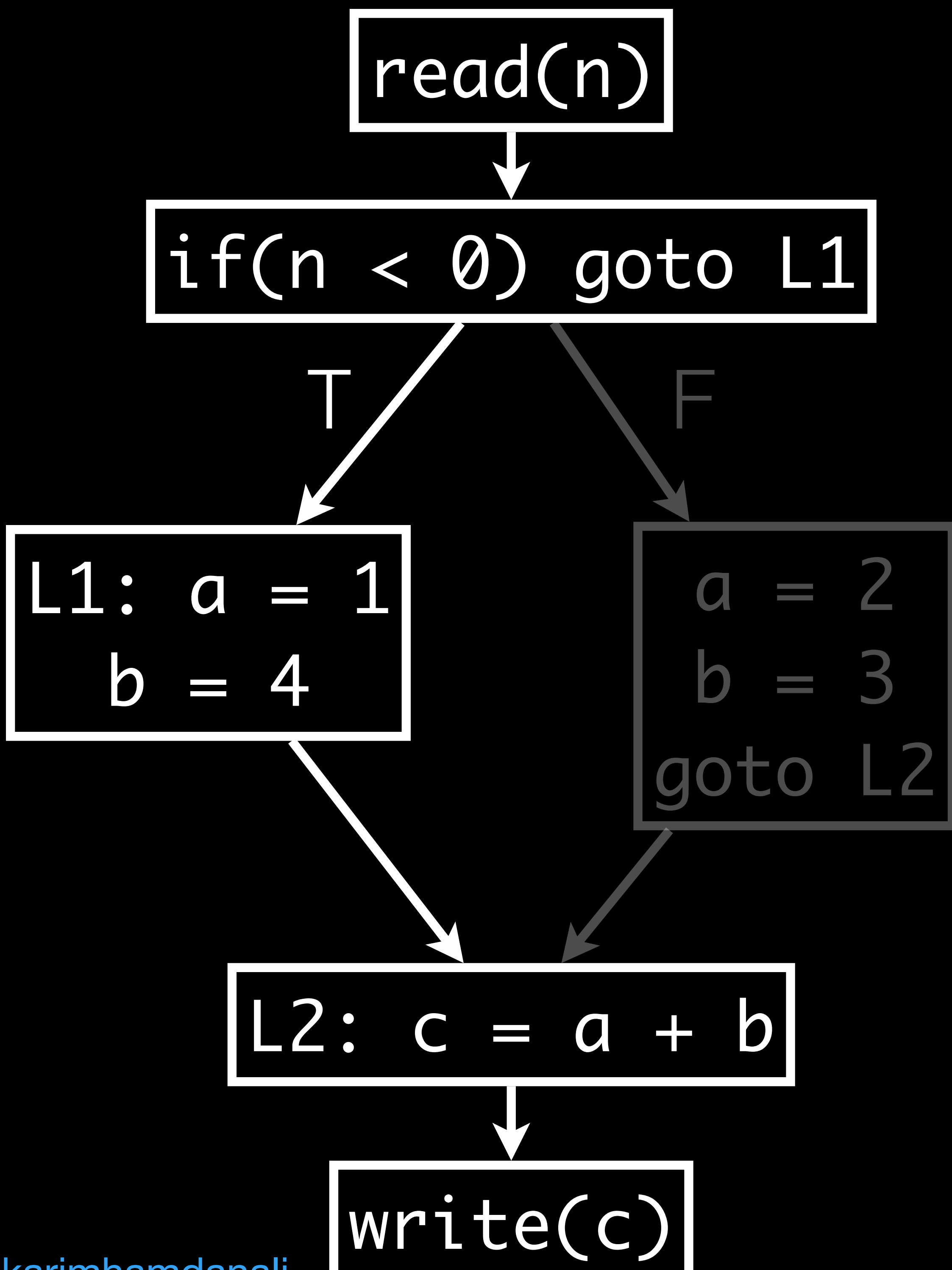

```
read(n);

if(n < 0) {
  a = 1;
  b = 4;
} else {
  a = 2;
  b = 3;
}

c = a + b;
write(c);
```

Basic-Block Graph

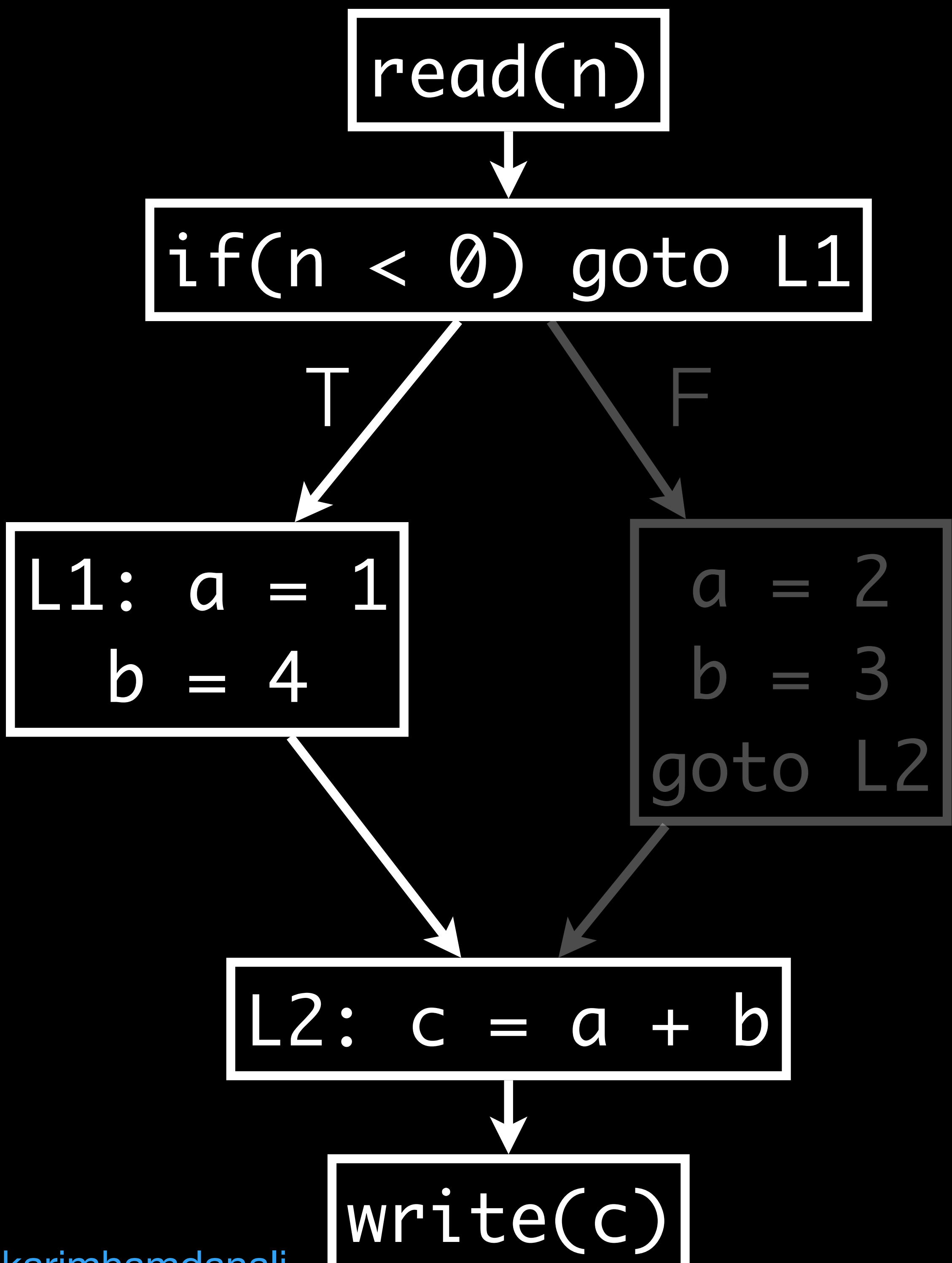



```
read(n);

if(n < 0) {
  a = 1;
  b = 4;
} else {
  a = 2;
  b = 3;
}

c = a + b;
write(c);
```

A path

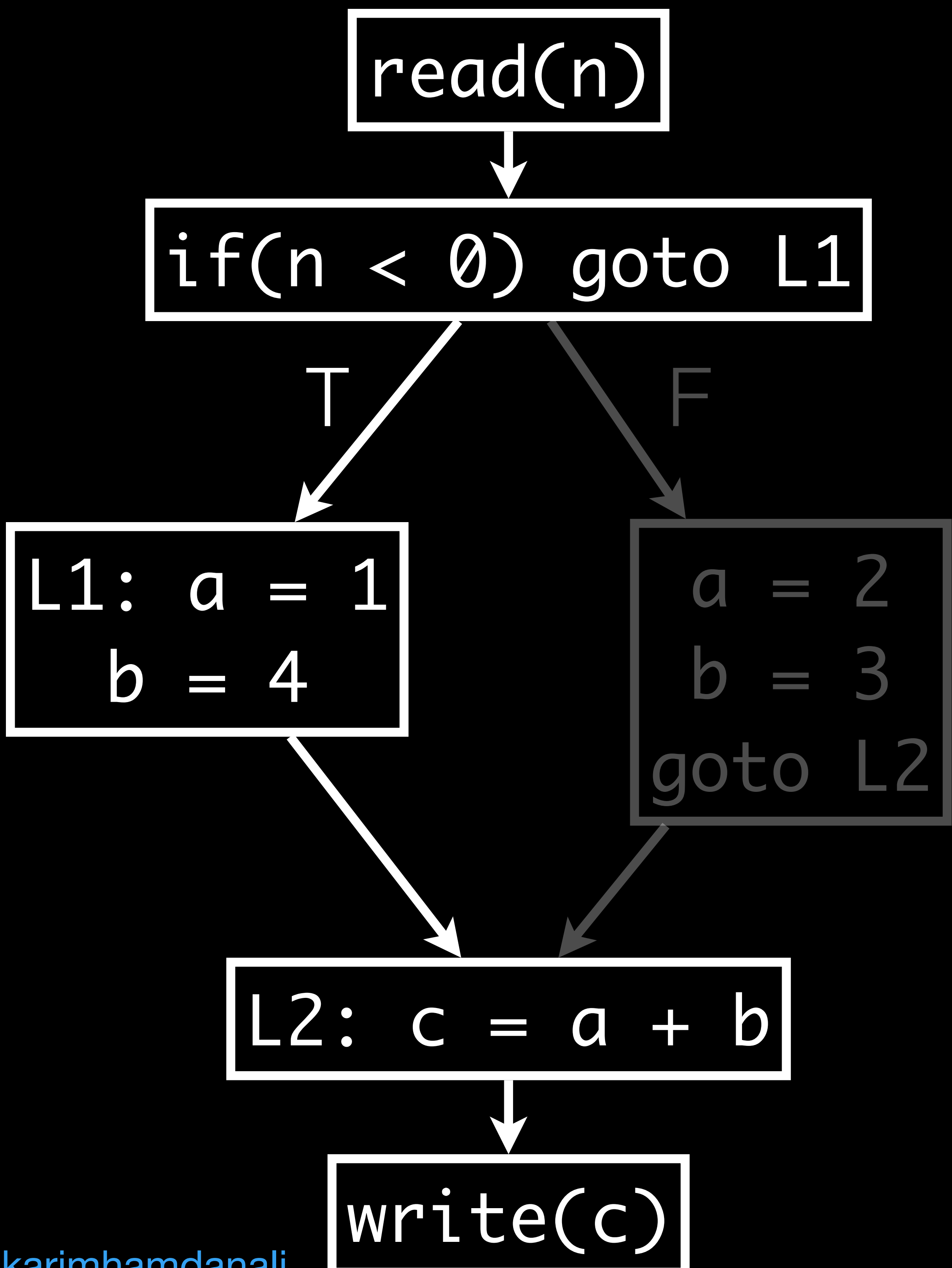


A path



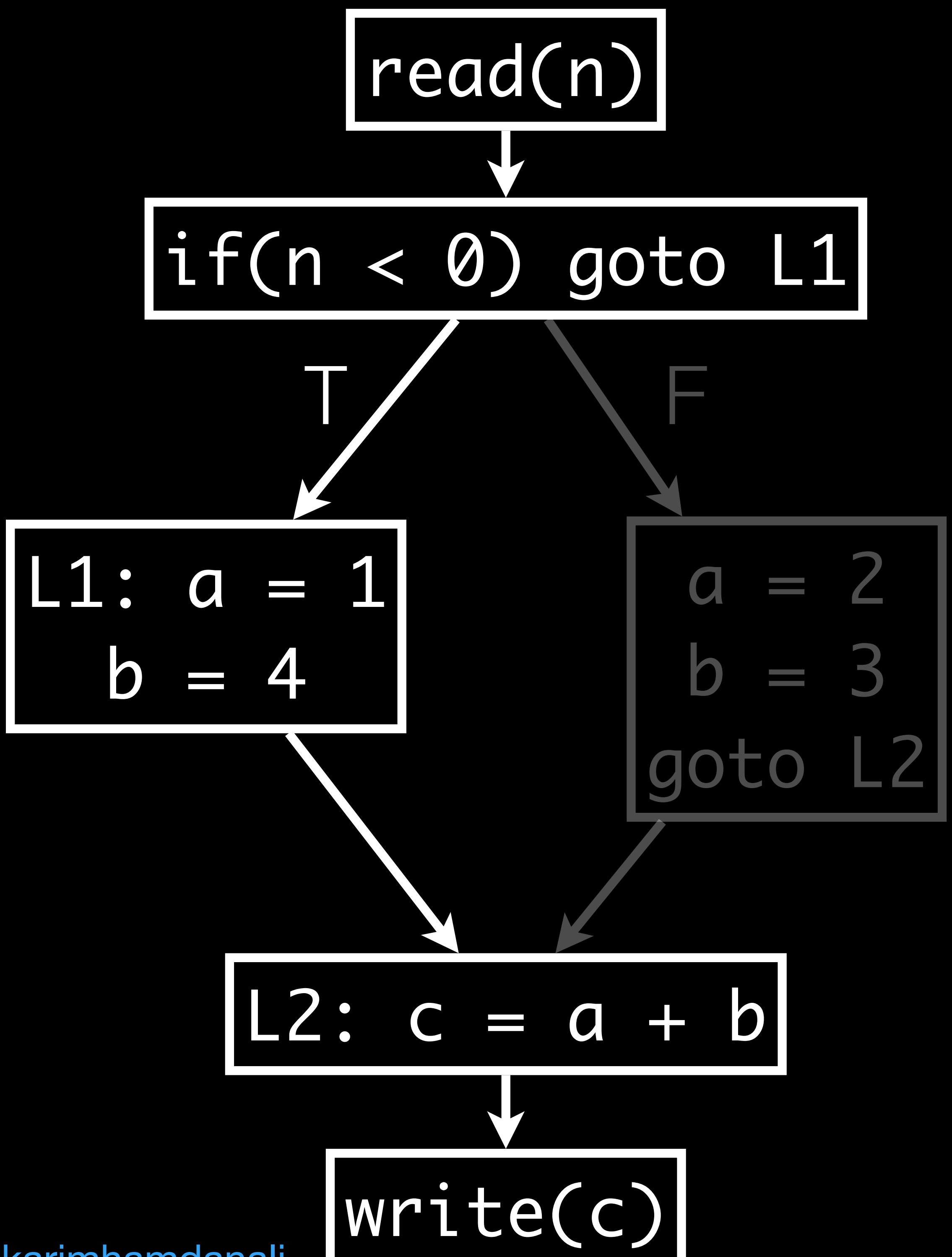
$f_{read(n)}(init)$

A path



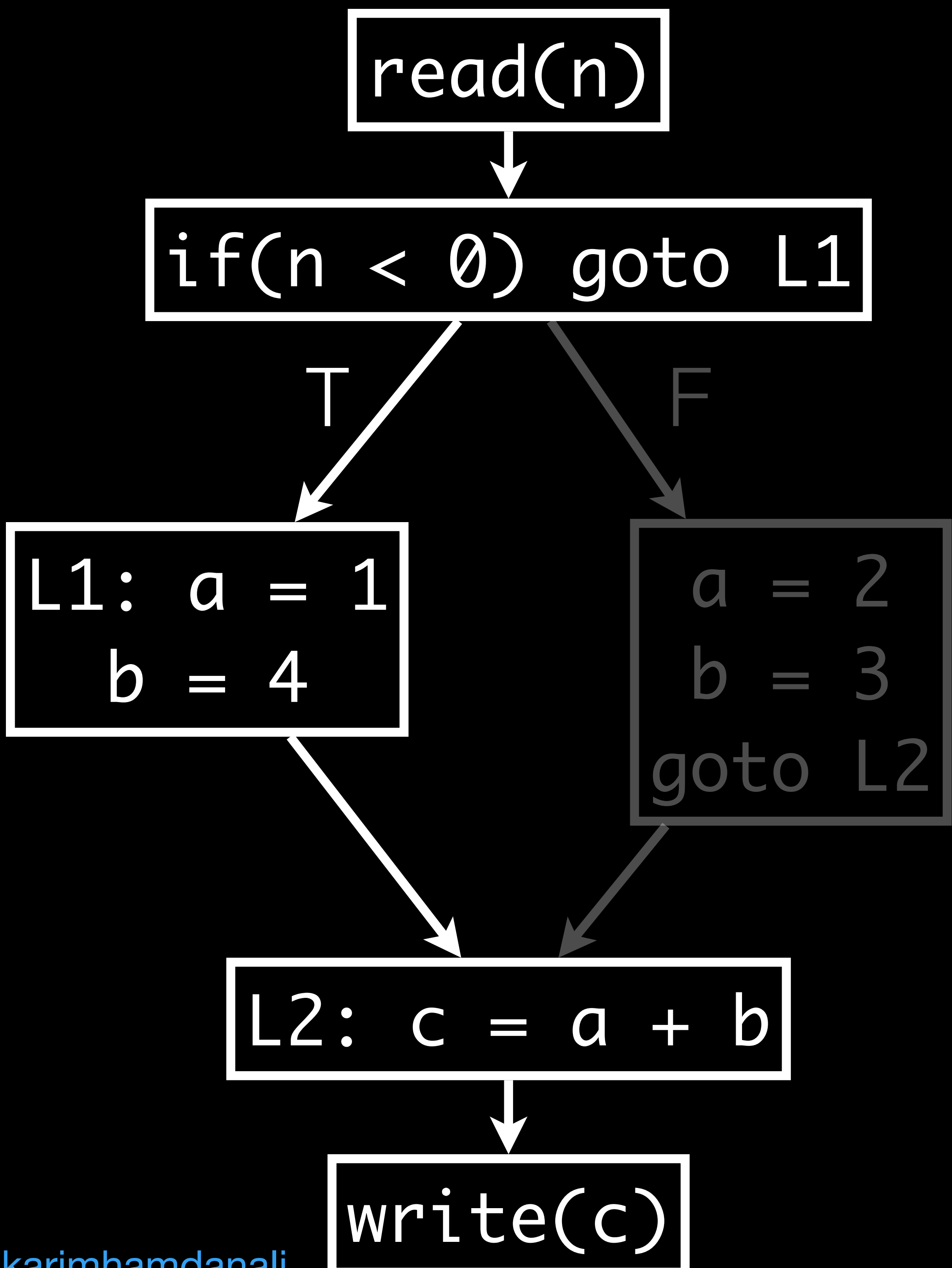
$$f_{n\,<\,0}(f_{read(n)}(init))$$

A path



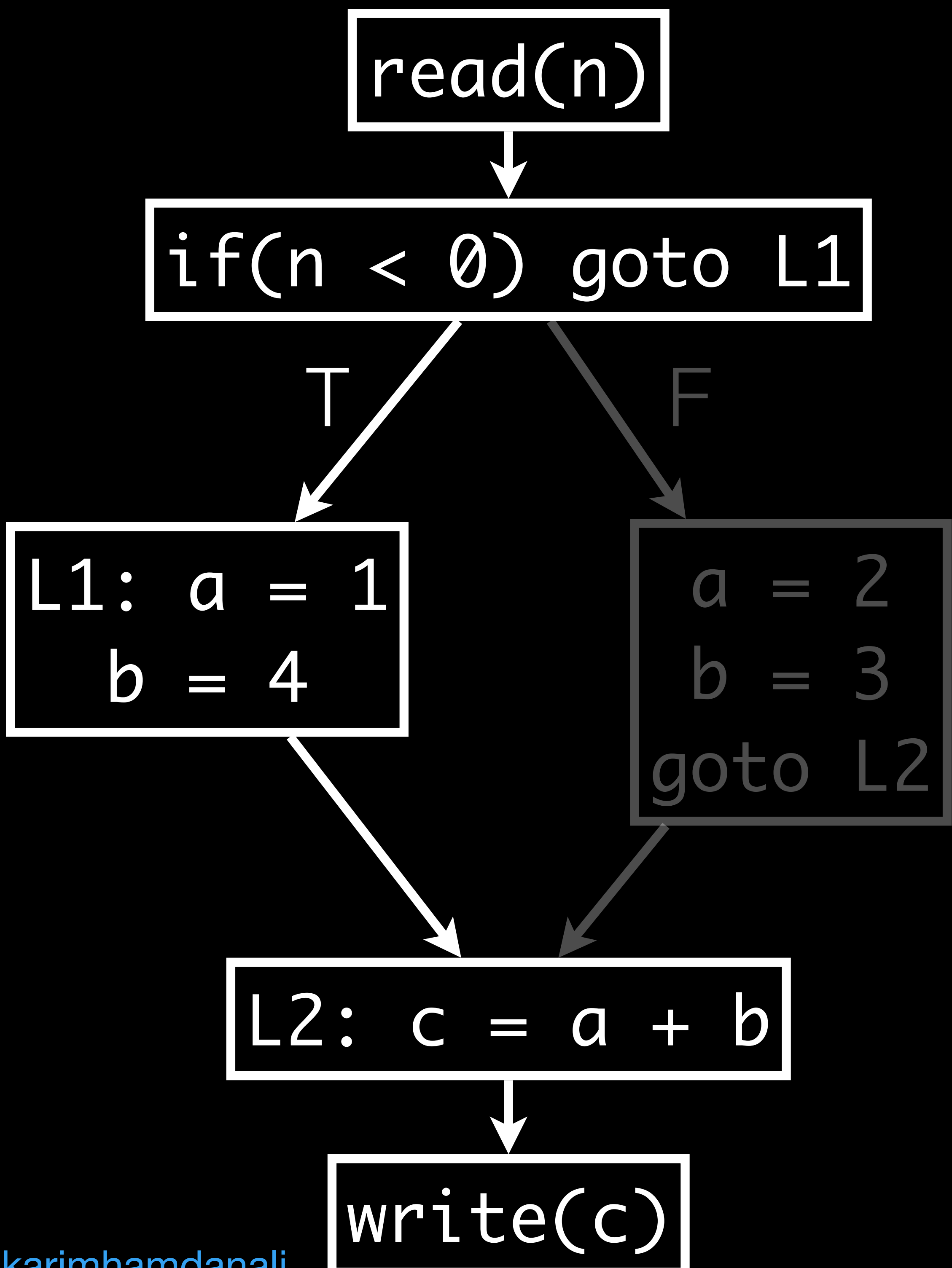
$f_{a\,=\,1}(f_{n\,<\,0}(f_{read(n)}(init)))$

A path



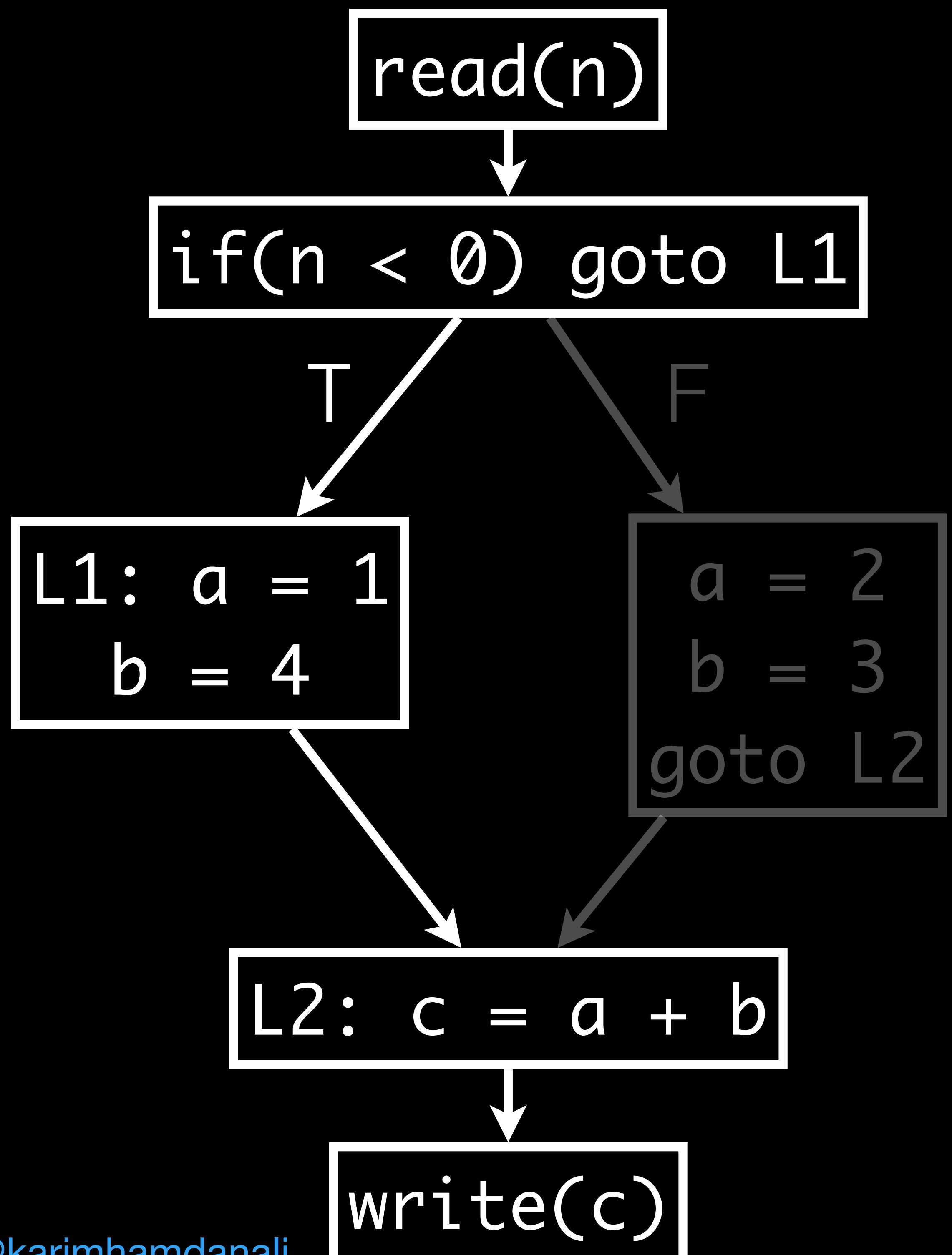
$$f_{b\,=\,4}(f_{a\,=\,1}(f_{n\,<\,0}(f_{\mathrm{read}(n)}(\mathrm{init}))))$$

A path



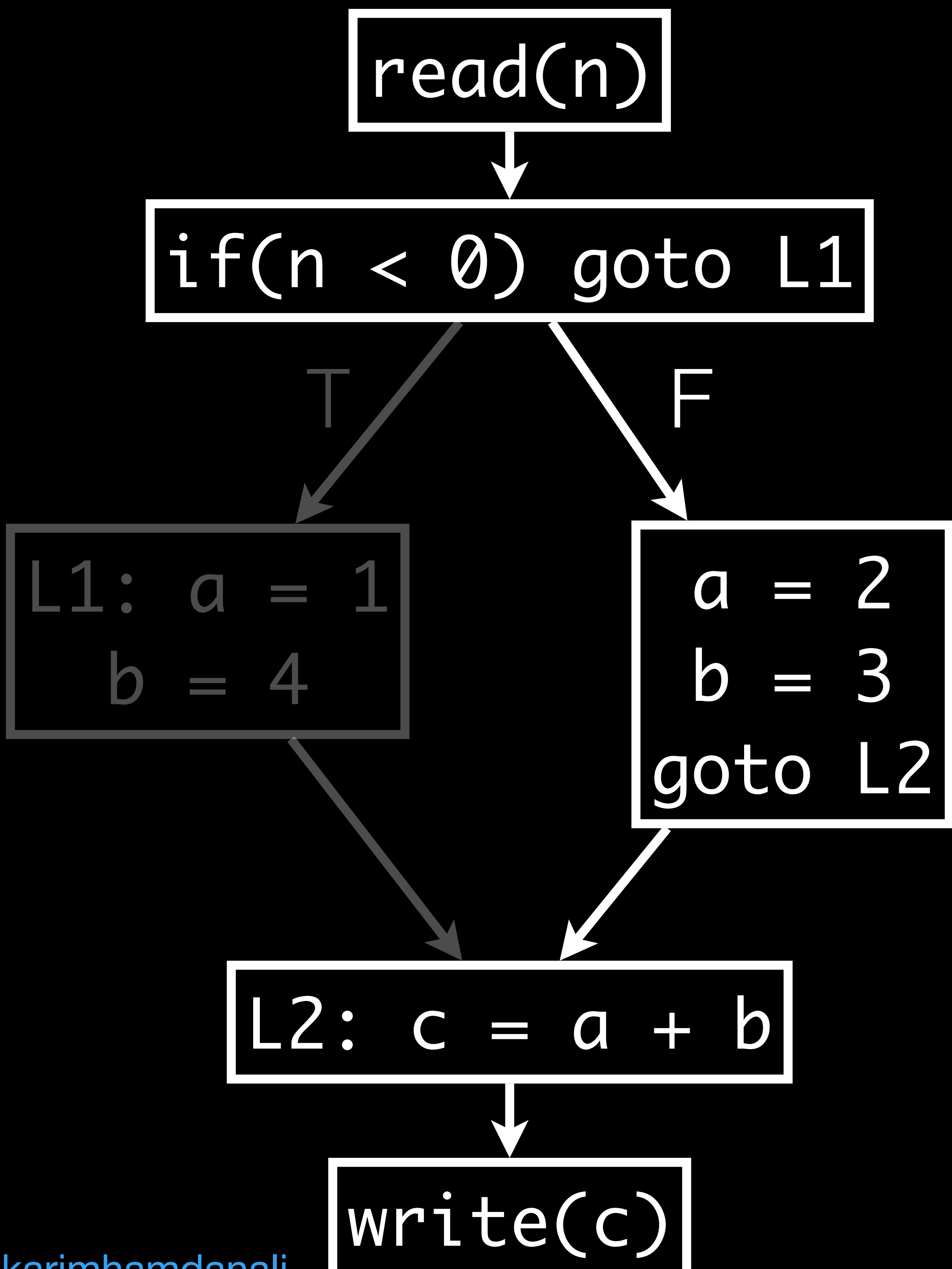
$$f_{c\,=\,a+b}(f_{b\,=\,4}(f_{a\,=\,1}(f_{n\,<\,0}(f_{read(n)}(init)))))$$

A path



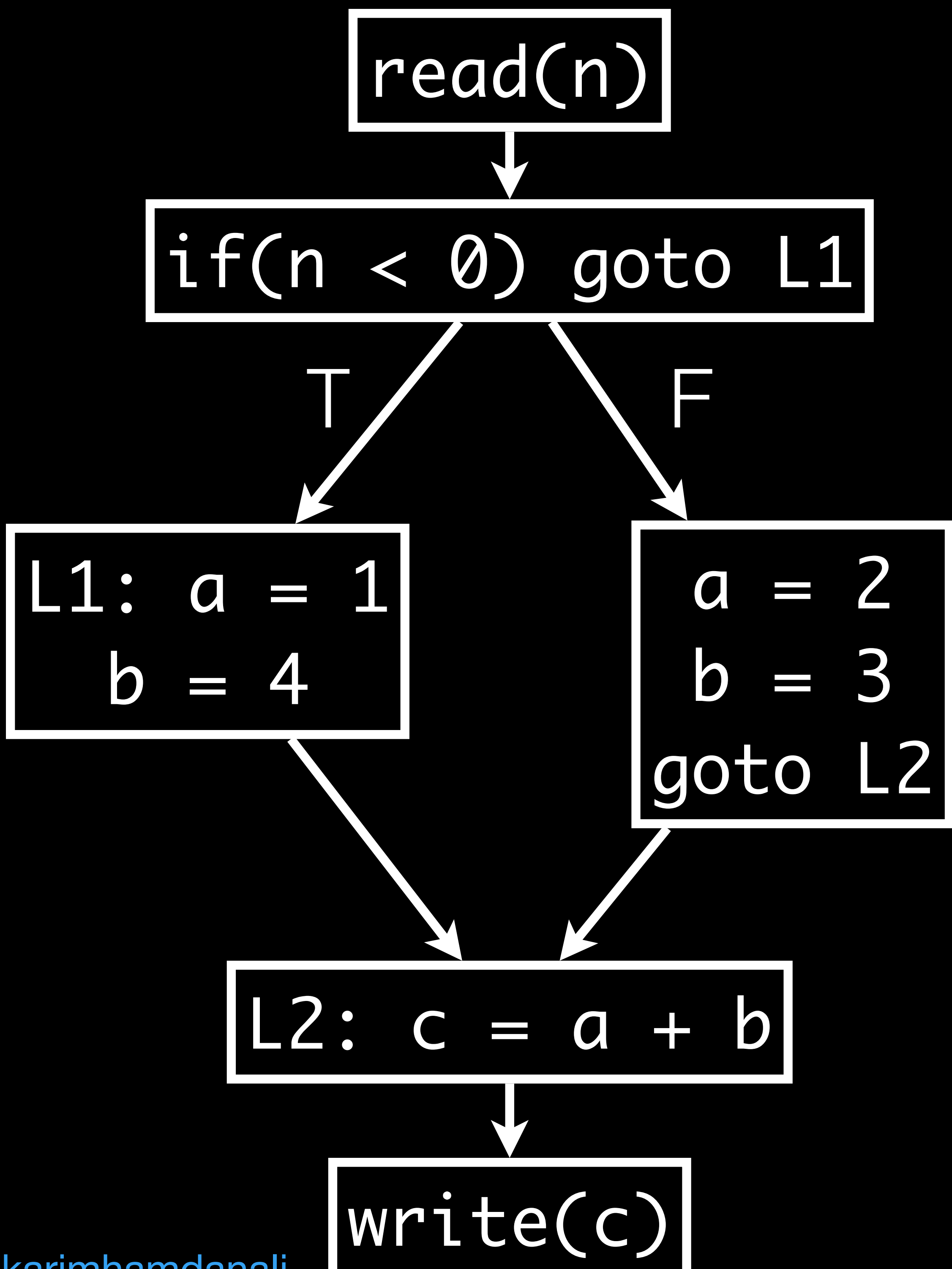
$$f_{write(c)}(f_{c\ =\ a+b}(f_{b\ =\ 4}(f_{a\ =\ 1}(f_{n\ <\ 0}(f_{read(n)}(init))))))$$

Another path



$$f_{\mathrm{write(c)}}(f_{c\,=\,a+b}(f_{b\,=\,3}(f_{a\,=\,2}(f_{n\,<\,0}(f_{\mathrm{read(n)}}(\mathrm{init}))))))$$

Paths Summary



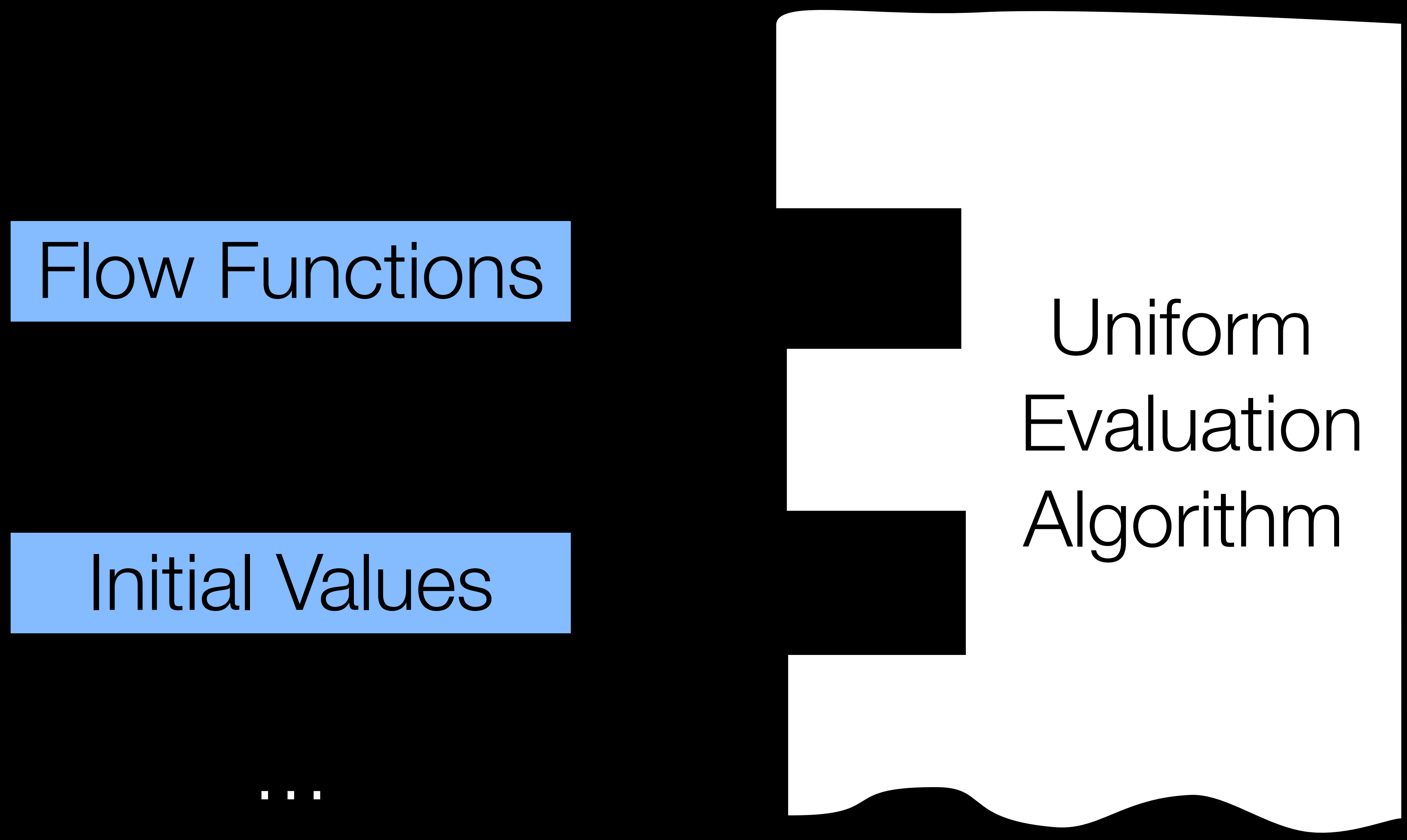
$$f_{write(c)}(f_{c\,=\,a+b}(f_{b\,=\,4}(f_{a\,=\,1}(f_{n\,<\,0}(f_{read(n)}(init))))))$$

$$\sqcup$$

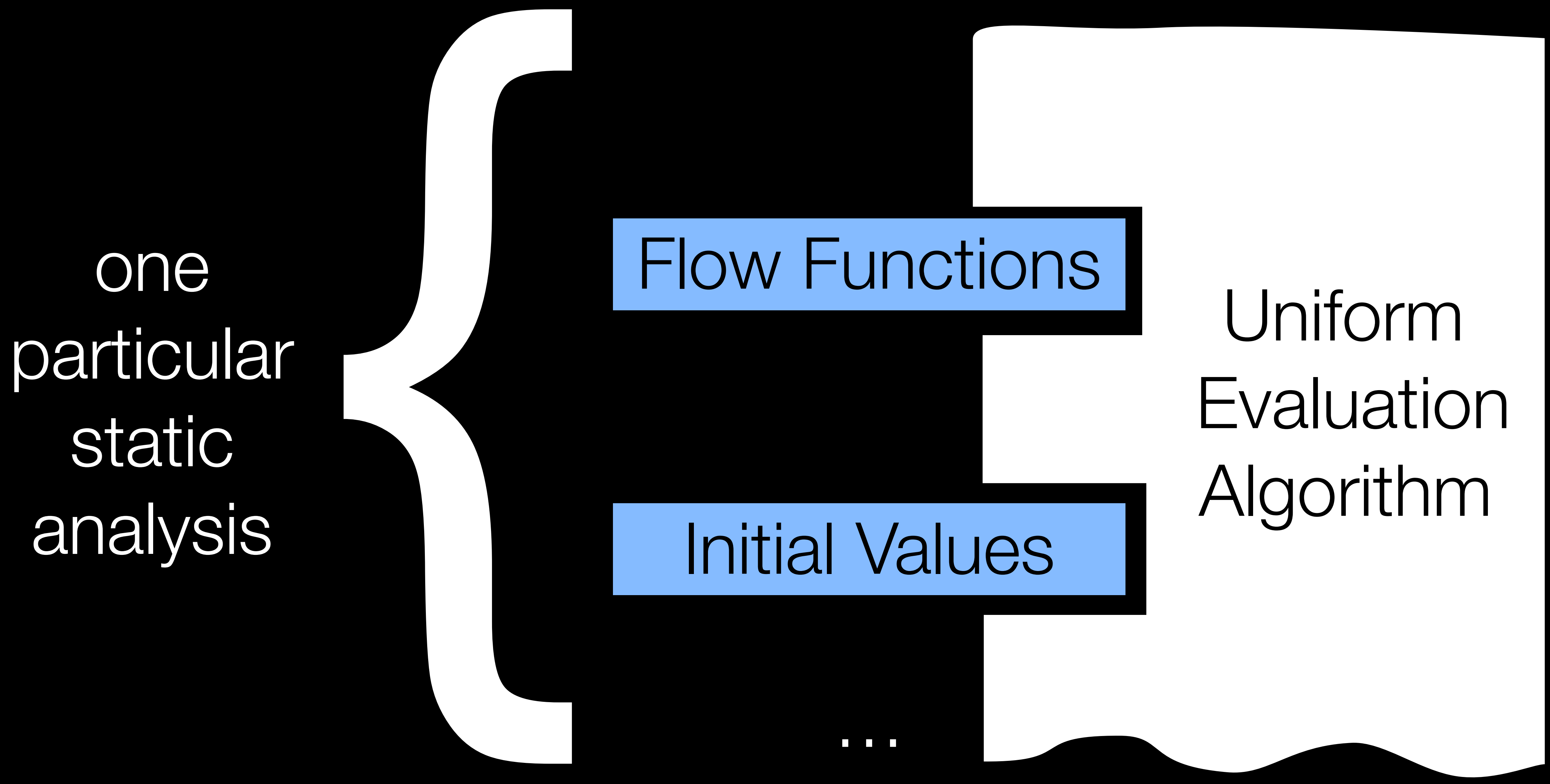
$$f_{write(c)}(f_{c\,=\,a+b}(f_{b\,=\,3}(f_{a\,=\,2}(f_{n\,<\,0}(f_{read(n)}(init))))))$$

Computable Solution: Monotone Framework

Monotone Framework



Monotone Framework



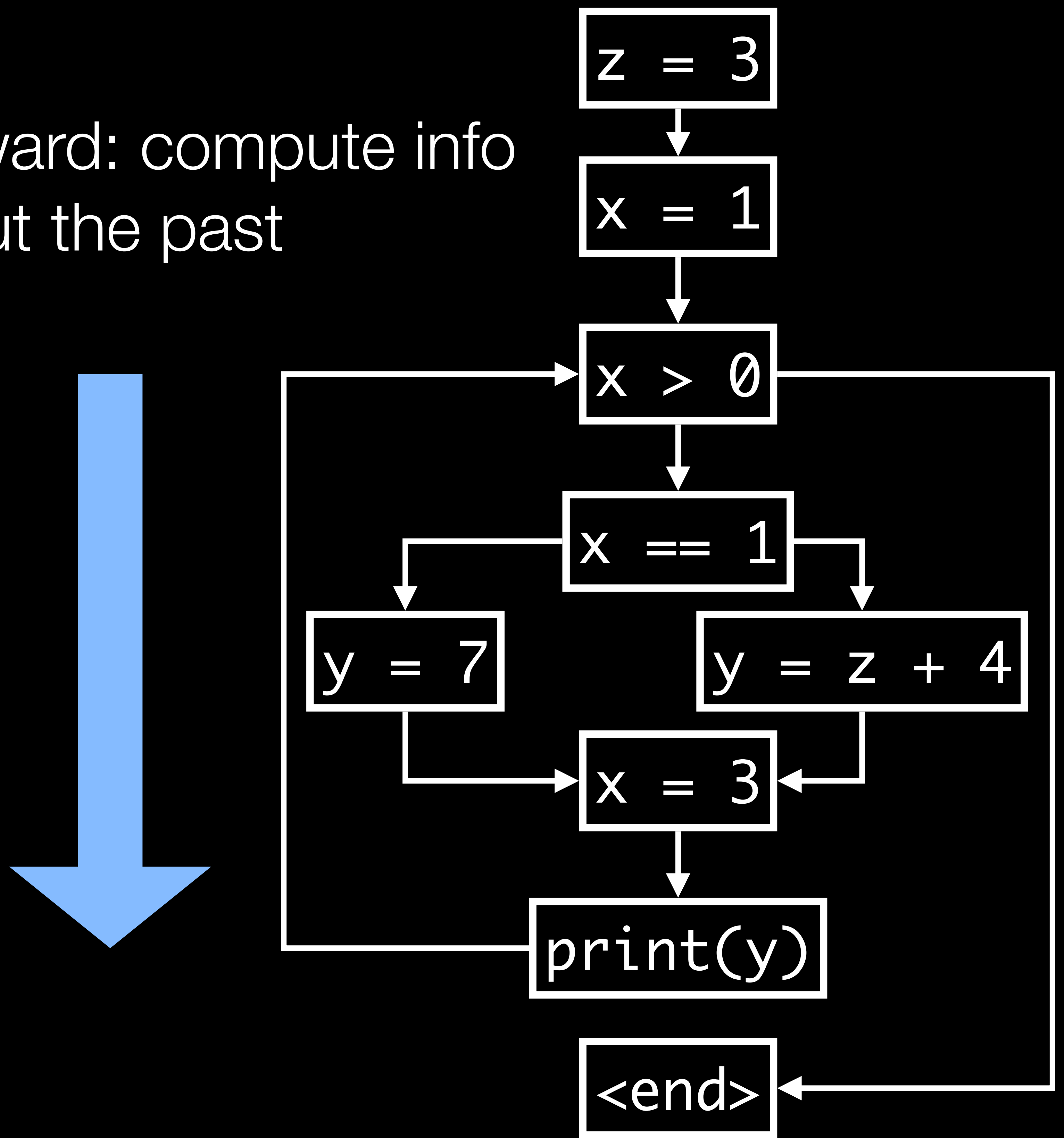
Monotone Framework

Parameter	Type
Forward or Backward	Boolean
Analysis Abstraction	Lattice
Effect of Each Statement on Info	Set of Flow Functions
Initialization	Lattice Values
Merge Operator	Binary Operator on Lattice Values

1. Forward or Backward

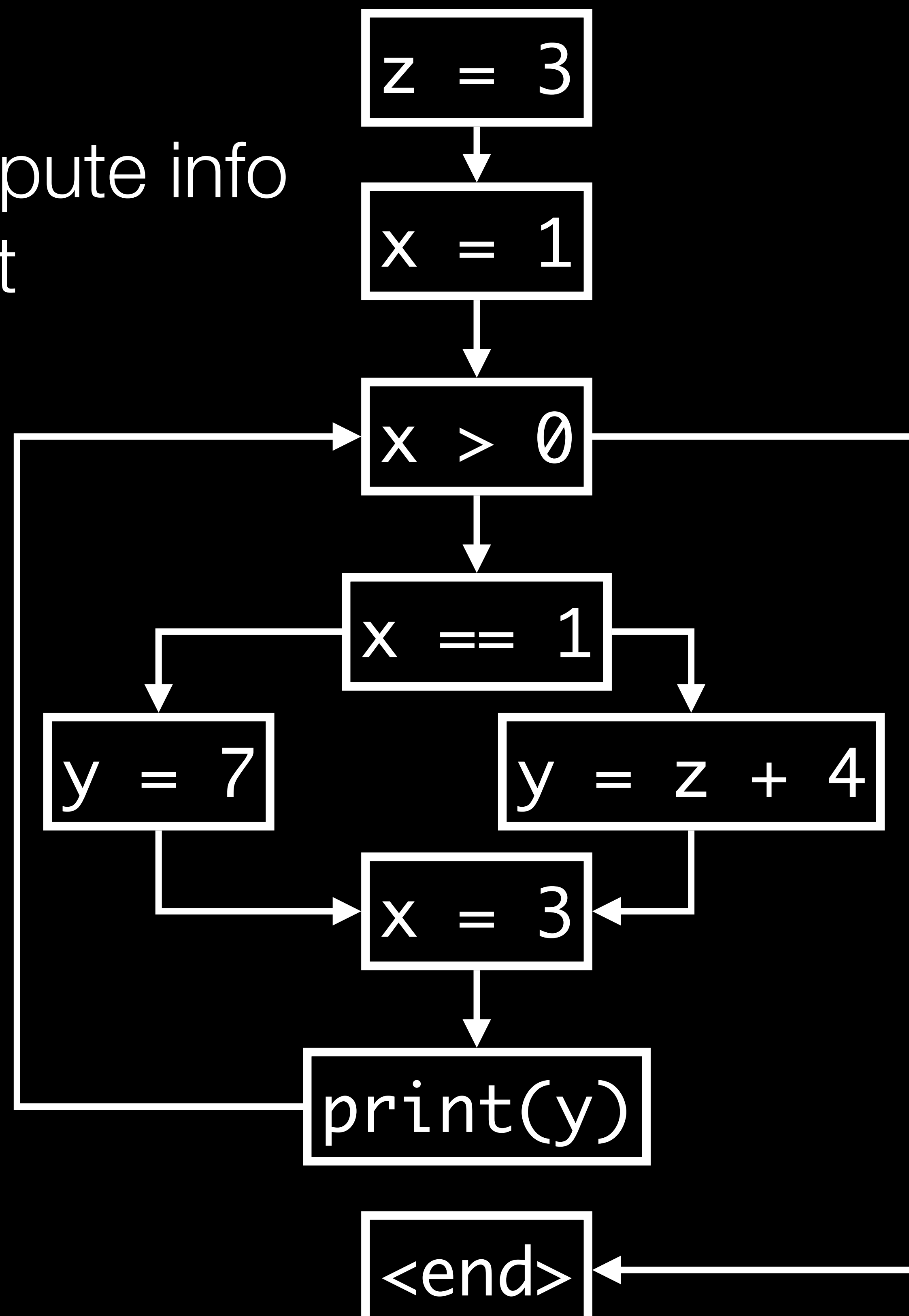
1. Forward or Backward

Forward: compute info about the past



1. Forward or Backward

Forward: compute info about the past



Backward: compute info about the future

1. Forward or Backward

Analysis	Name	Direction
Which values does a variable carry?	Constant Propagation	Forward
Which variables will still be used?	Live Variables	Backward
Will this file handle be properly close?	Typestate	Backward
Has a variable been defined?	Possibly Defined Variables	Forward

1. Forward or Backward

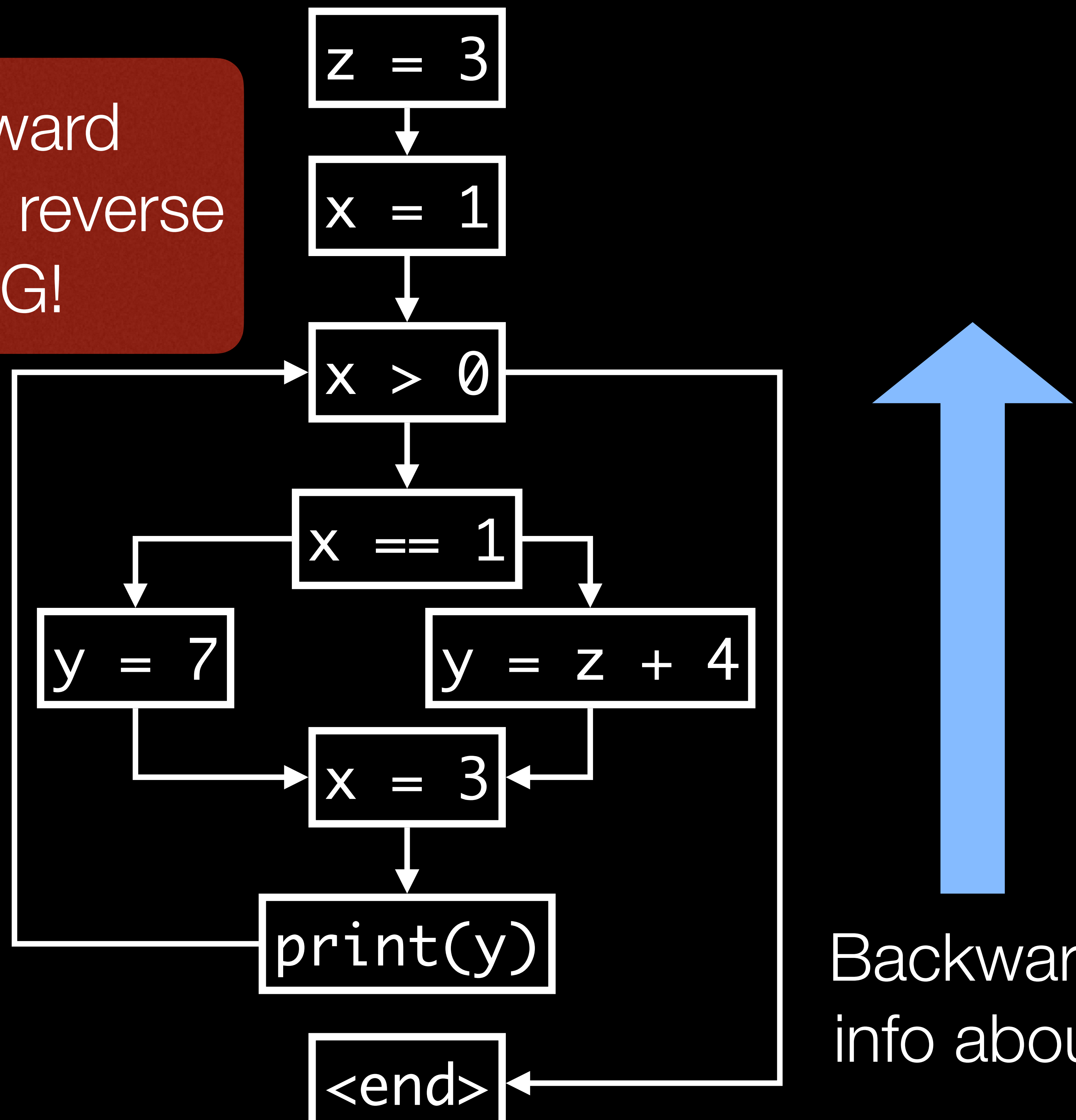
Forward: Did p2 ever hold the value in password?

```
p = password();

p2 = p;

print(p2);
```

Backward: Can the password be printed in the future?

1. Forward or Backward

== Forward analysis on reverse of CFG!

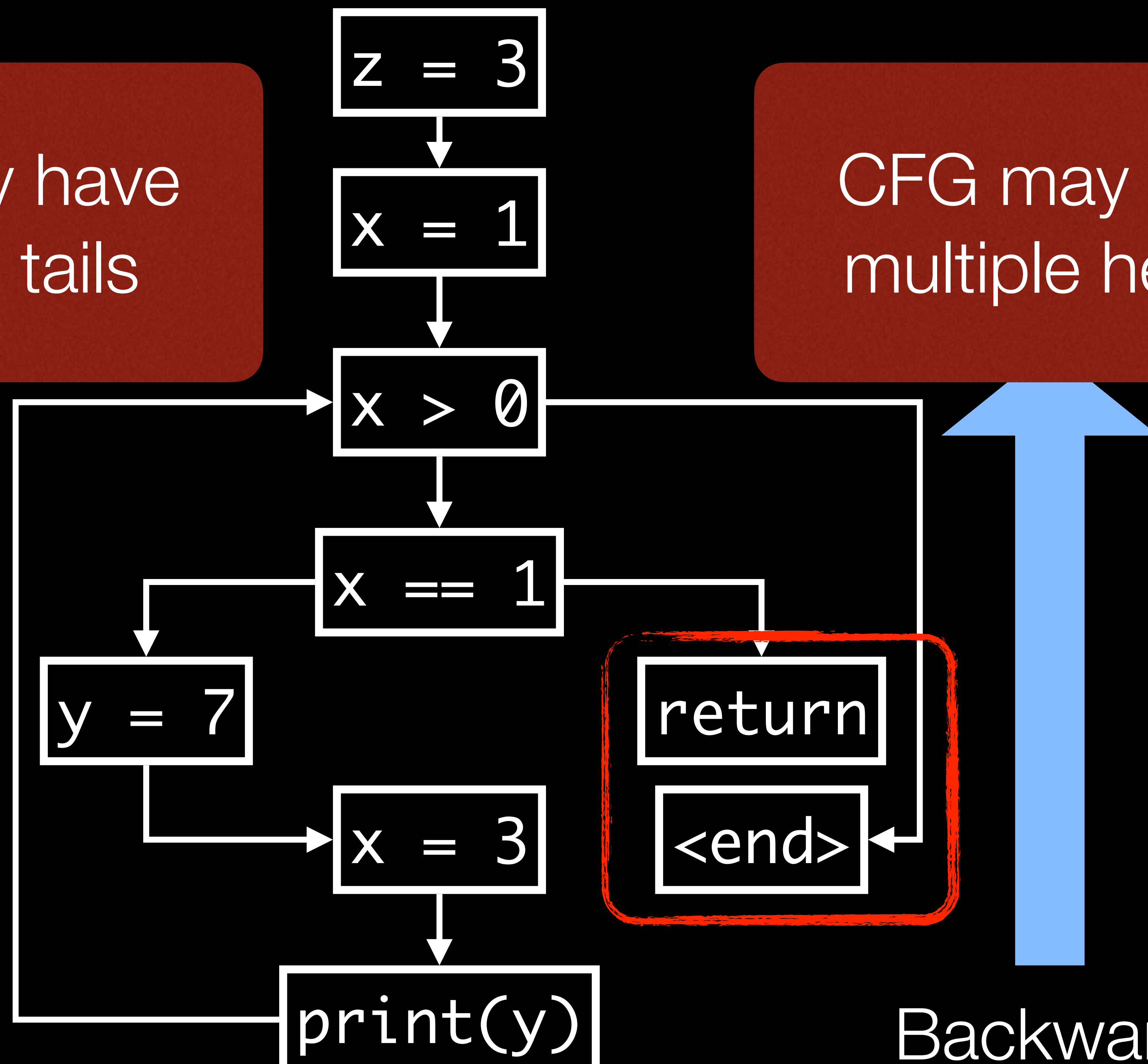


Backward: compute info about the future

1. Forward or Backward

CFG may have multiple tails

CFG may have multiple heads



Backward: compute info about the future

2. Analysis Abstraction

2. Analysis Abstraction

Lattice depends heavily on analysis problem!

2. Analysis Abstraction

Example: Constant Propagation

What is the constant value of `x` at location `s`?

{ }

```
a = 1
```

{a↦1}

```
b = 2
```

{a↦1, b↦2}

```
o = null
o = new O()
o.foo(a, b)
```

2. Analysis Abstraction

Example: Nullness Analysis

Which variable is null
at location `s`?

```
        { }
a = 1
  ↓     { }
b = 2
  ↓     { }
o = null
  ↓     {o}
o = new O()
  ↓     { }
o.foo(a, b)
```

2. Analysis Abstraction

Example: Type Analysis

Which runtime type could reference variable `x` at location `s`?

{ }

```
a = 1
```

{ }

```
b = 2
```

{ }

```
o = null
```

{o↦NullType}

```
o = new O()
```

{o↦O}

```
o.foo(a, b)
```

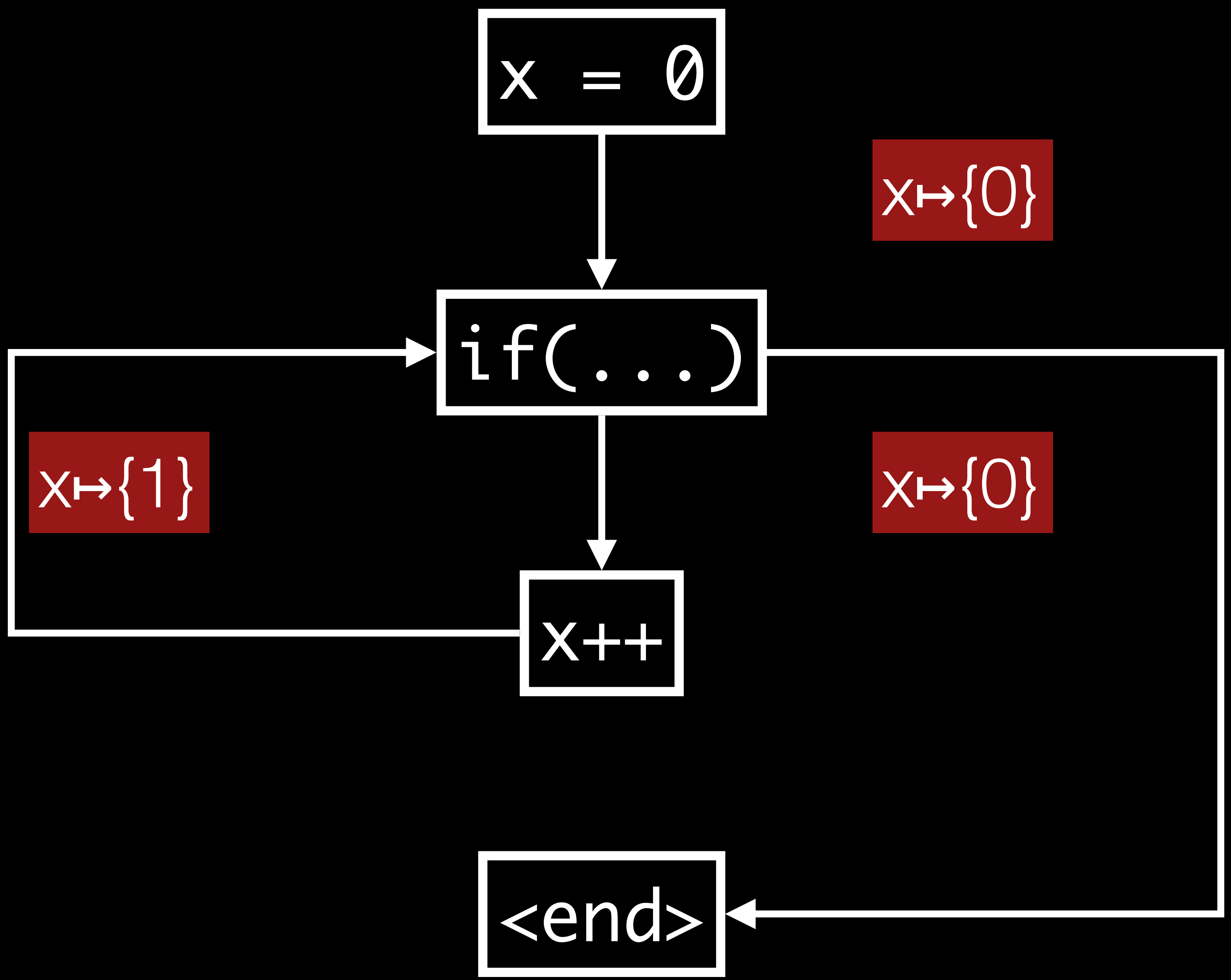
2. Analysis Abstraction

Lattice are "often"
sets of "something"

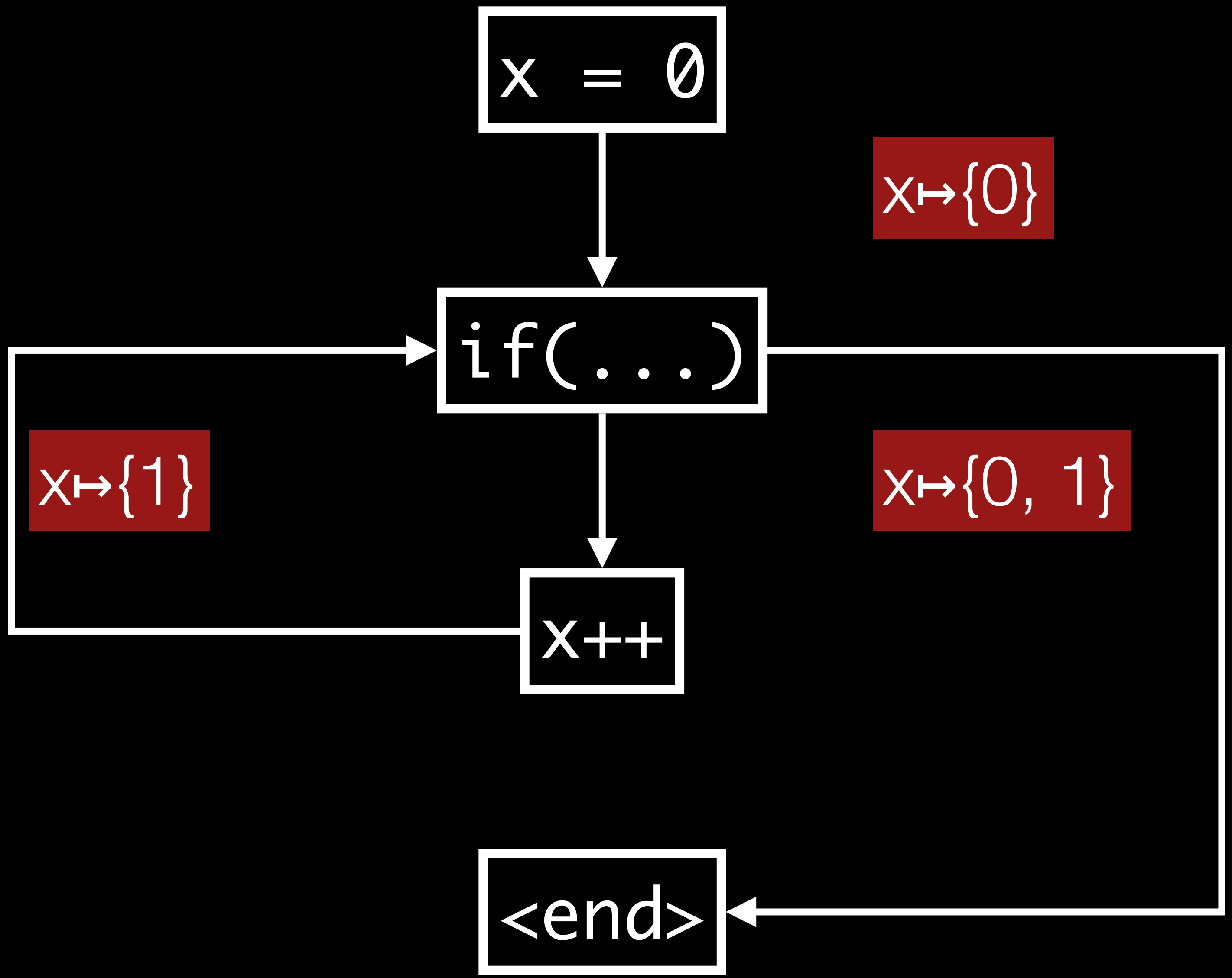
2. Analysis Abstraction

… but why do we need a lattice?

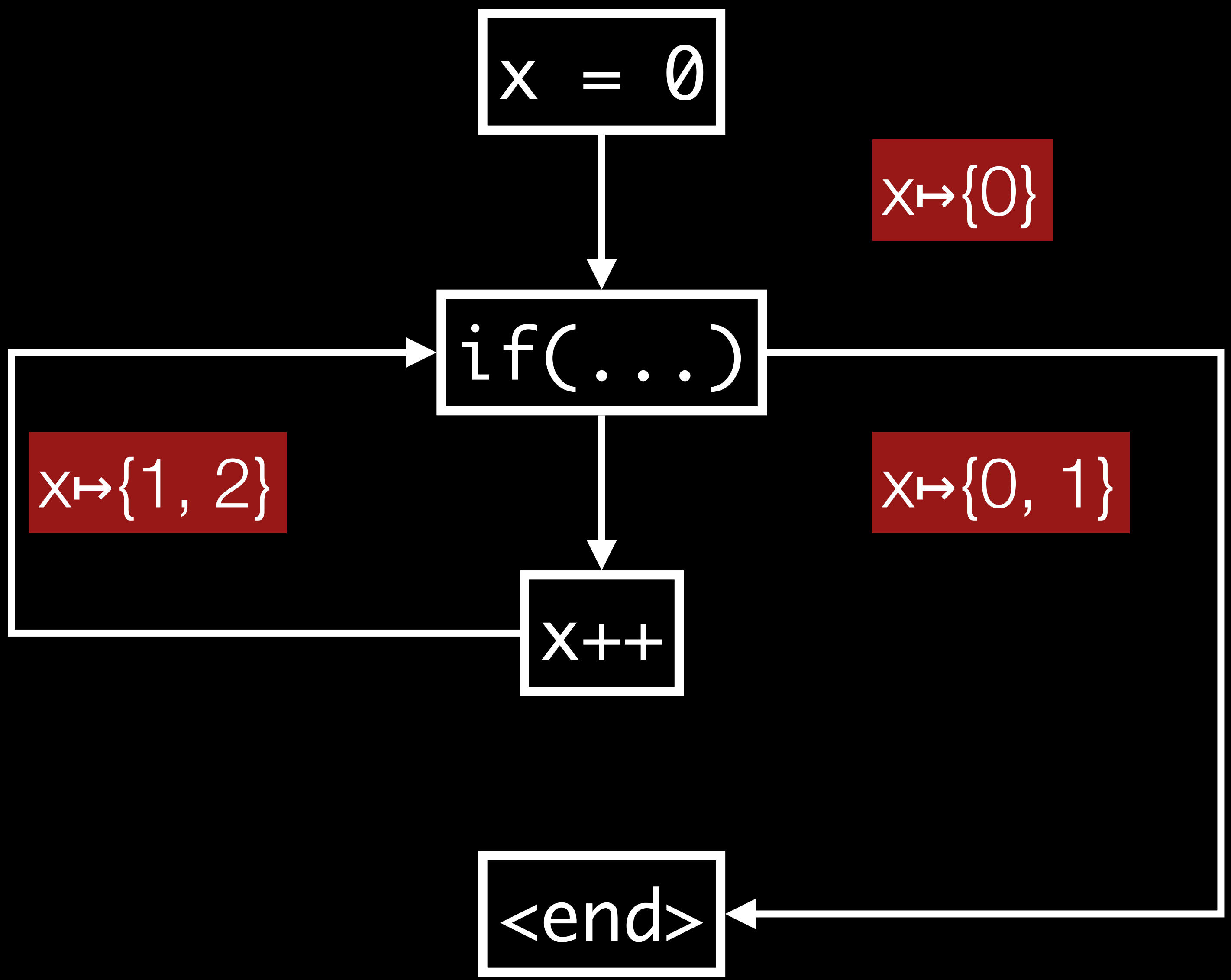
2. Analysis Abstraction
Why do we need a lattice?



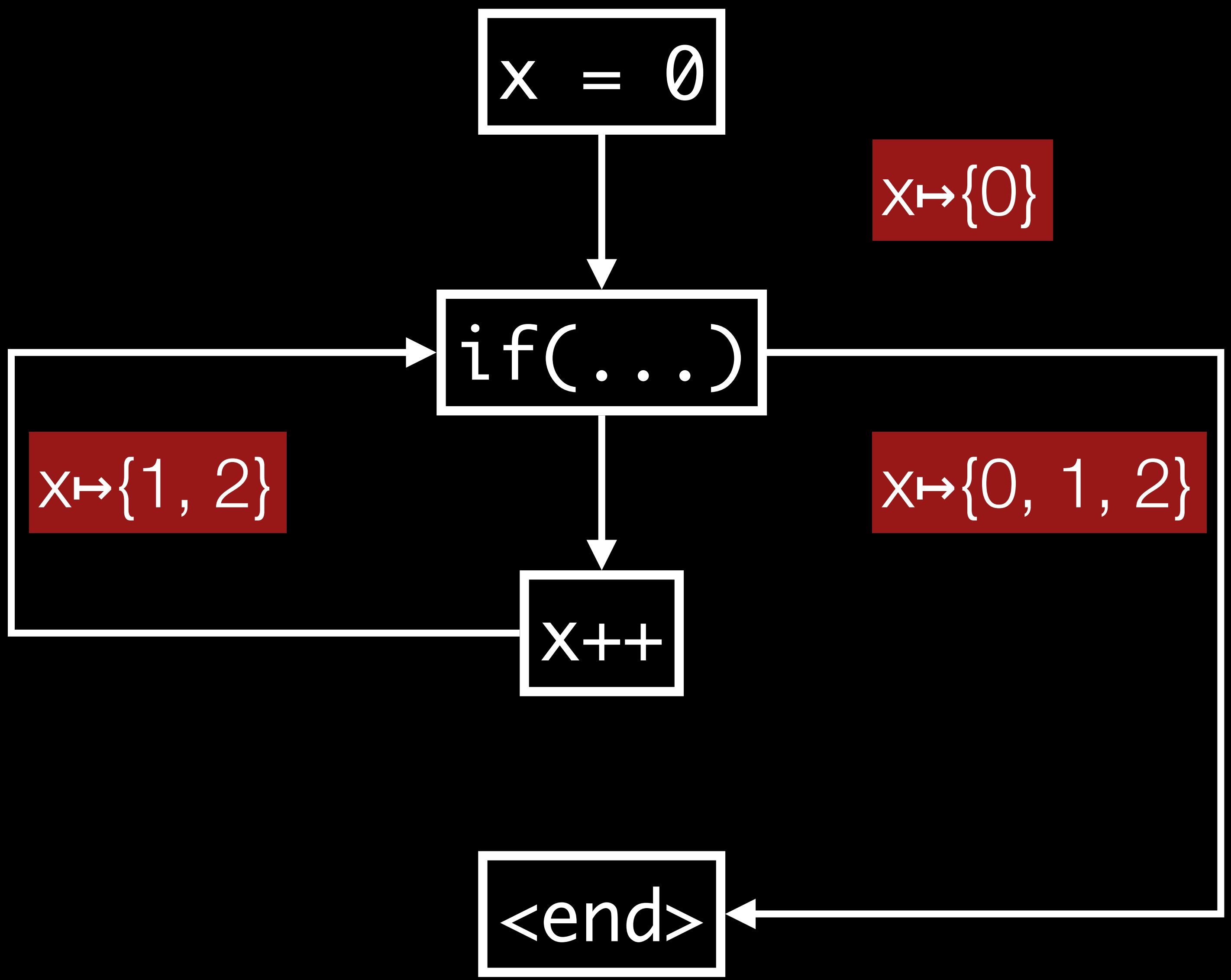
2. Analysis Abstraction
Why do we need a lattice?



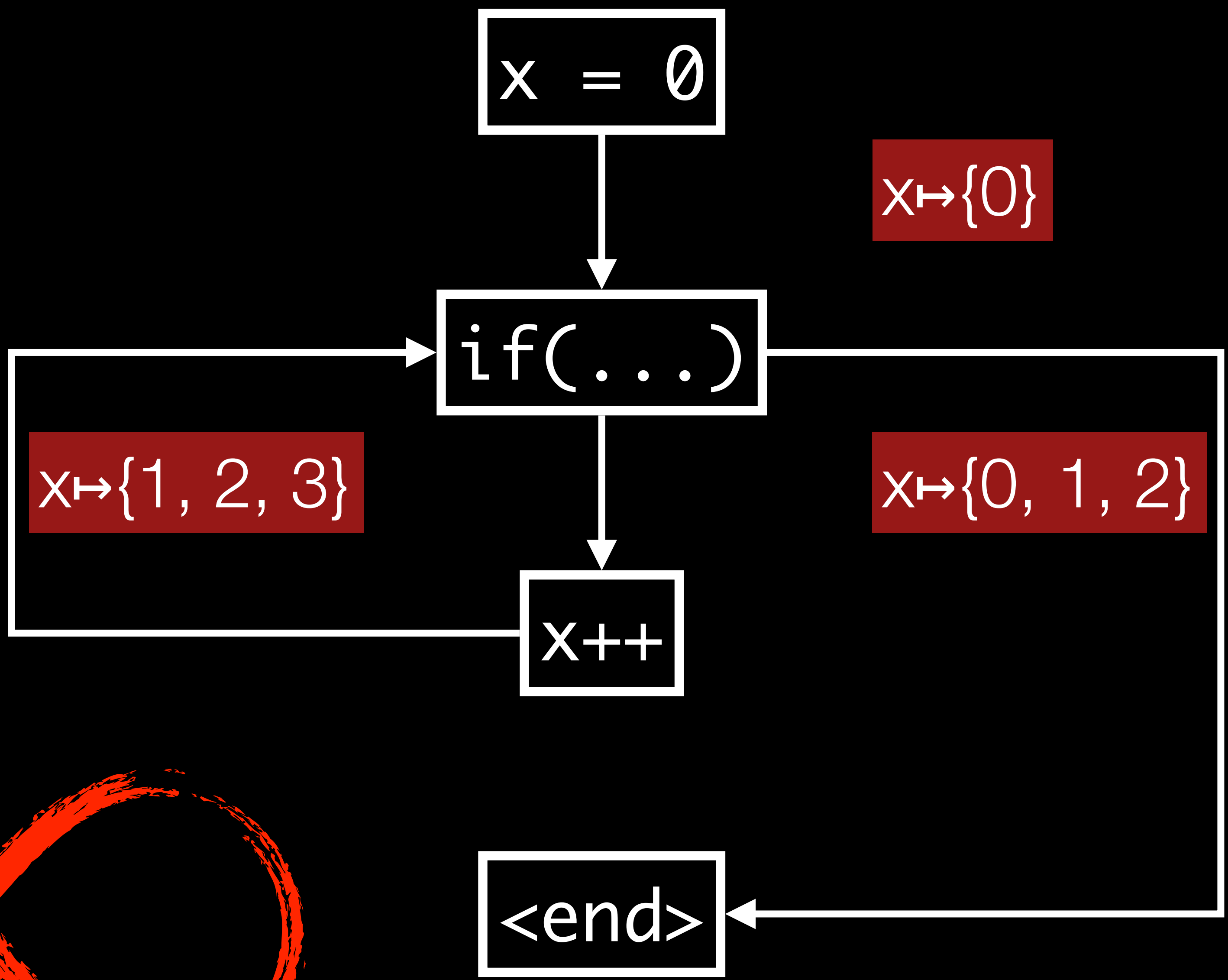
2. Analysis Abstraction
Why do we need a lattice?



2. Analysis Abstraction
Why do we need a lattice?



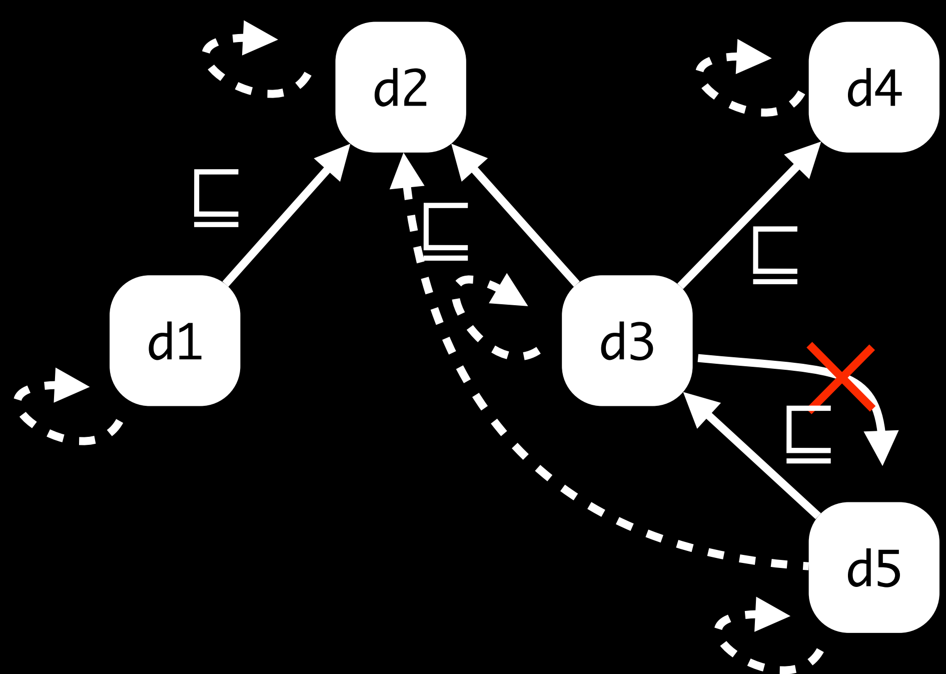
2. Analysis Abstraction
Why do we need a lattice?



2. Analysis Abstraction
Partially-Ordered Set (poset)

- If U is a set and $\sqsubseteq$ is a binary relation on U, then the system $(U, \sqsubseteq)$ is a poset if:
    - $\forall\, x \in U : x \sqsubseteq x$ ($\sqsubseteq$ is reflexive)
    - $\forall\, x, y, z \in U : (x \sqsubseteq y \wedge y \sqsubseteq z) \implies x \sqsubseteq z$ ($\sqsubseteq$ is transitive)
    - $\forall\, x, y, z \in U : (x \sqsubseteq y \wedge y \sqsubseteq x) \implies x == y$ ($\sqsubseteq$ is anti-symmetric)

$x \sqsubseteq y$ means:
y is a **safe approximation** of *x*, or **at least as sound as** *x*



2. Analysis Abstraction
Partially-Ordered Set (poset)

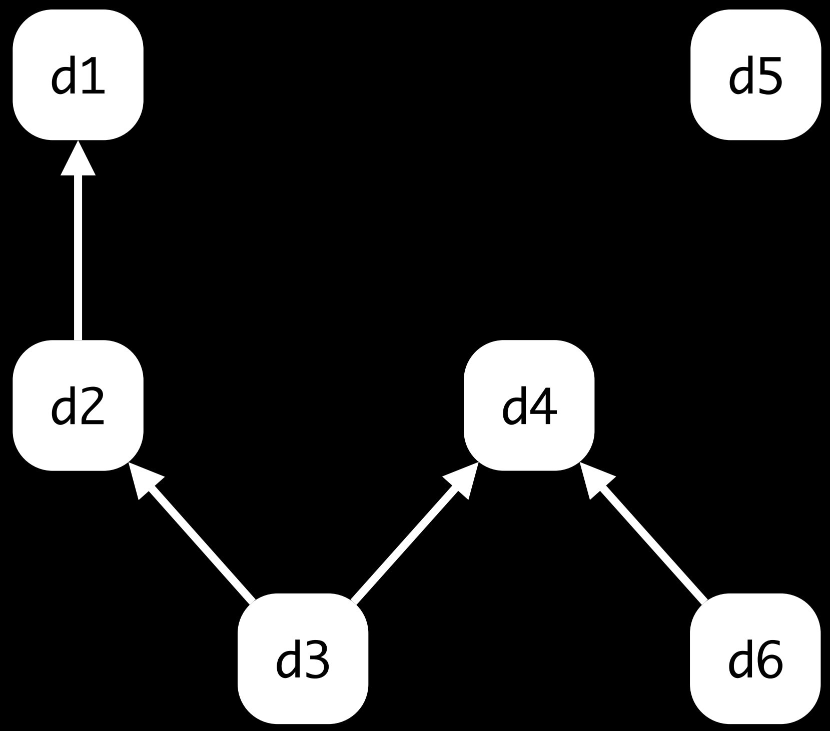
- Examples
  - $\leq$ over natural numbers
  - $\subseteq$ over finite sets

Key Takeaway About Posets

- A poset is a set with a notion of "less than or equal"

2. Analysis Abstraction
Aside: Hasse Diagrams

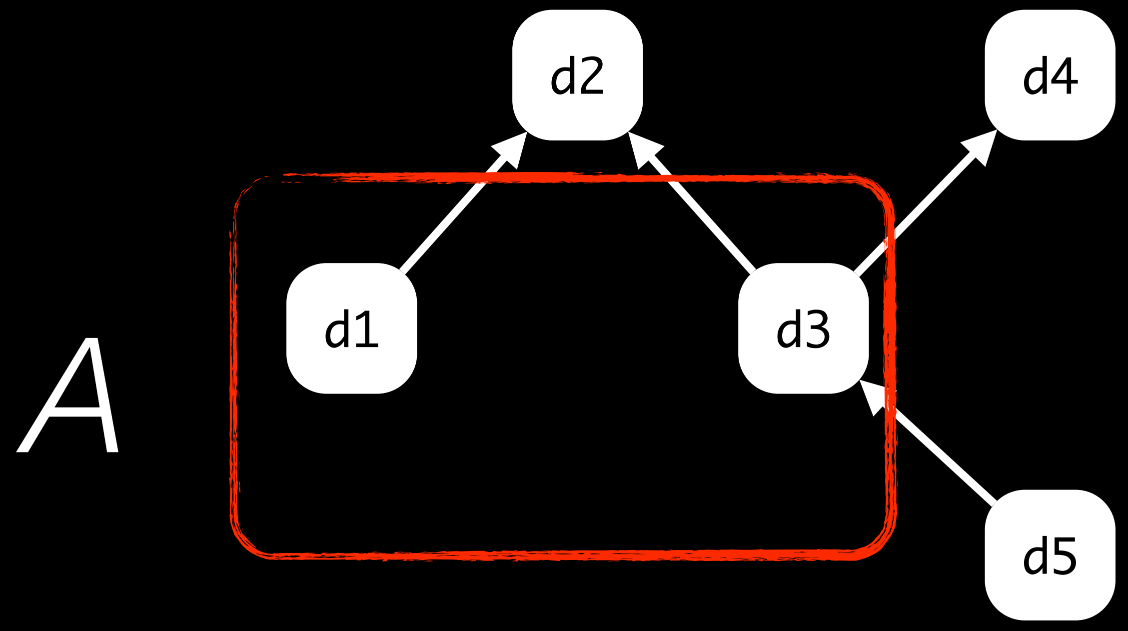
- Represent finite posets with a diagram
- Vertices represent elements of U
- Line from x to y if $x \sqsubseteq y$ and no z such that $x \sqsubseteq z \sqsubseteq y$
- Assume reflexivity
- Assume transitivity



Find all x,y such that $x \sqsubseteq y$

2. Analysis Abstraction
Upper Bound

- If (U, $\sqsubseteq$) is a poset and $A \subseteq$ U and $z \in$ U, then z is an upper bound of A if $\forall\, x \in A : x \sqsubseteq z$

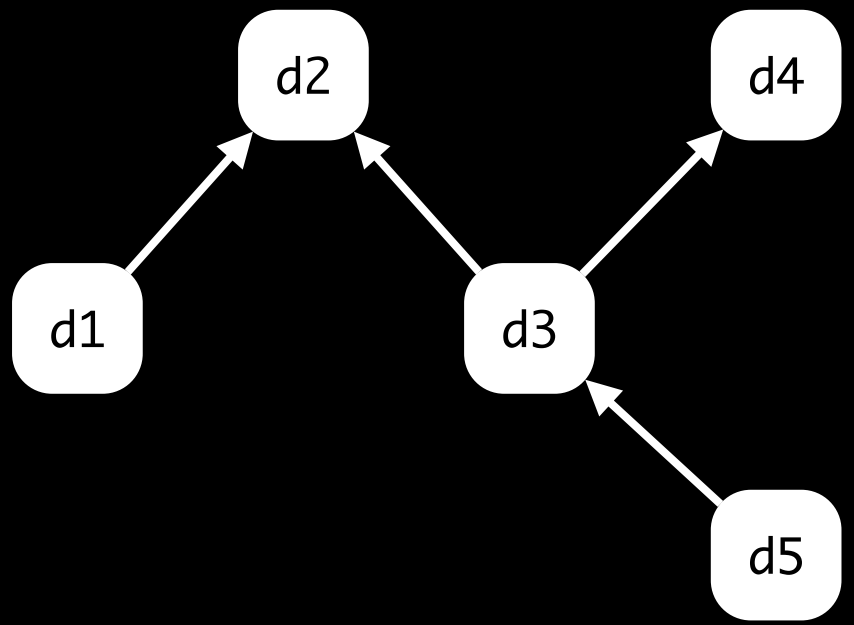


z is an **approximation** of every element of *A*

2. Analysis Abstraction
Upper Bound

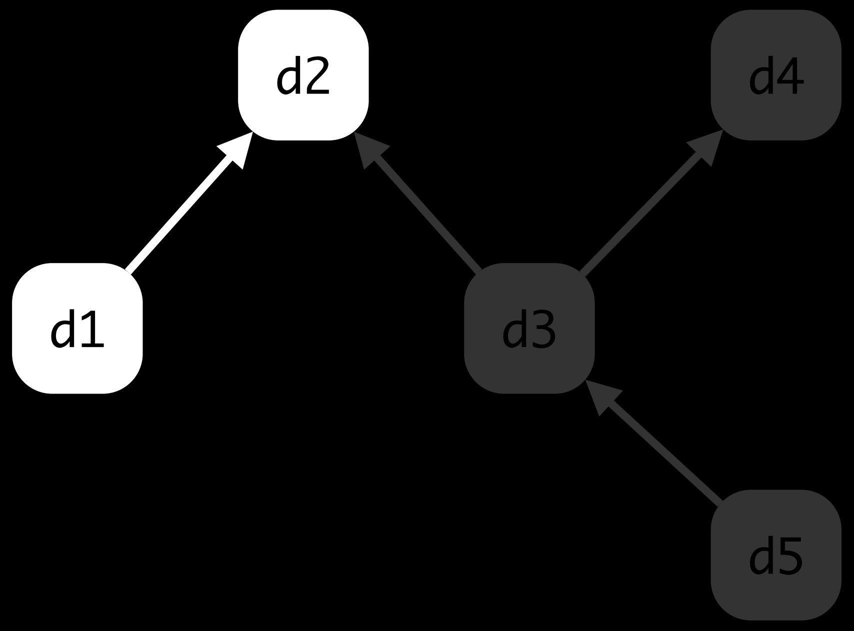
- If $(U, \sqsubseteq)$ is a poset and x, y, $z \in$ U, then z is an upper bound of x and y if $x \sqsubseteq z \wedge y \sqsubseteq z$

Definition specialized for $A = \{x,y\}$



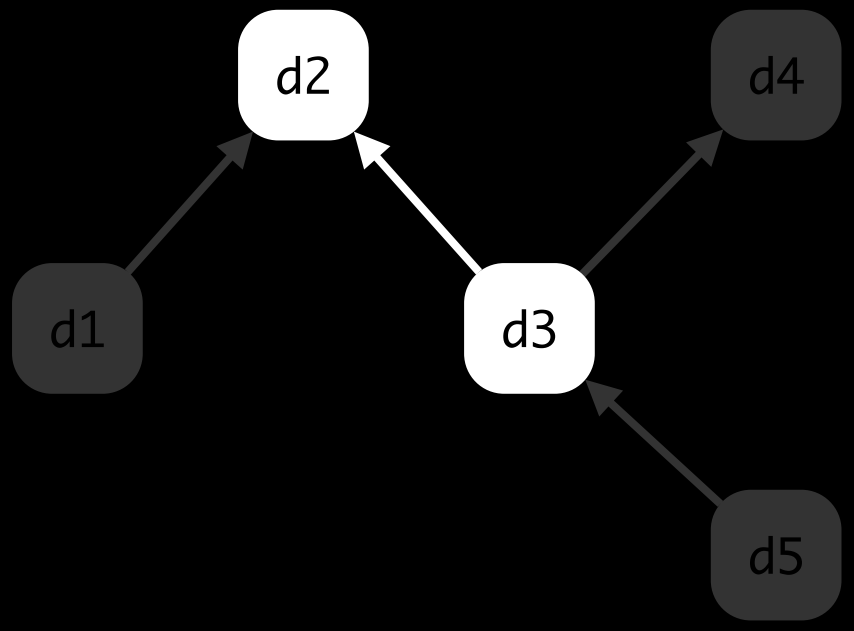
2. Analysis Abstraction
Upper Bound

- If (U, $\sqsubseteq$) is a poset and x, y, $z \in$ U, then z is an upper bound of x and y if $x \sqsubseteq z \land y \sqsubseteq z$



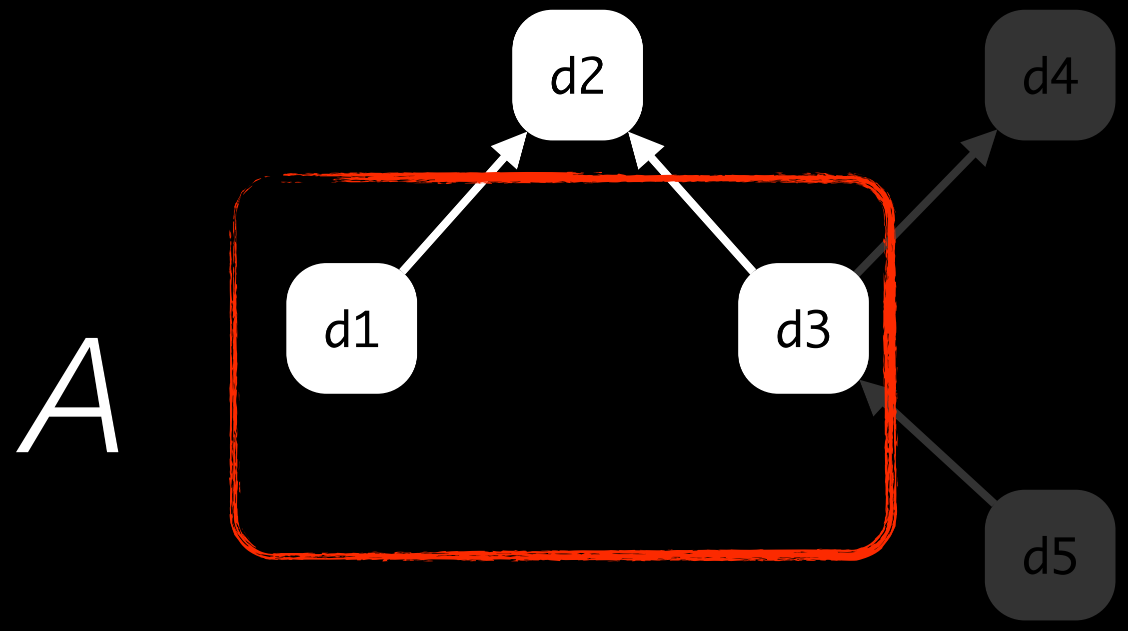
2. Analysis Abstraction
Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $x, y, z \in U$, then z is an upper bound of x and y if $x \sqsubseteq z \land y \sqsubseteq z$



2. Analysis Abstraction
Upper Bound

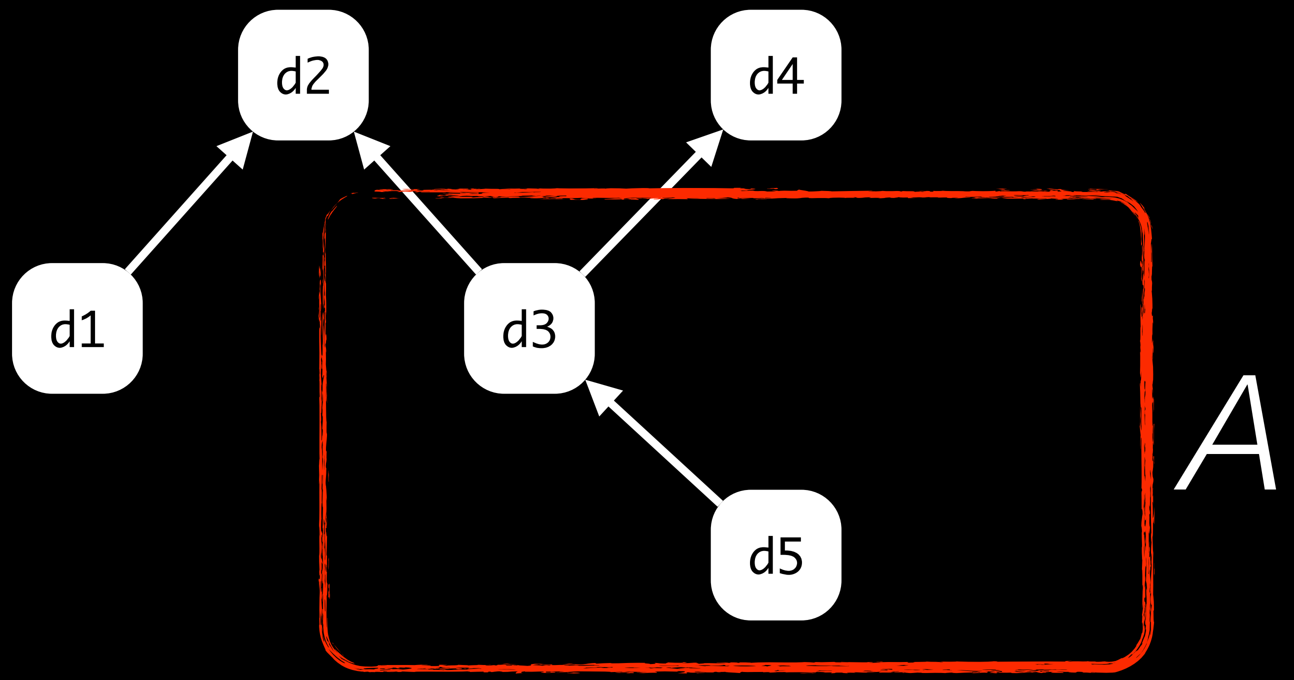
- If (U, $\sqsubseteq$) is a poset and x, y, $z \in$ U, then z is an upper bound of x and y if $x \sqsubseteq z \land y \sqsubseteq z$



d2 is an **approximation** of every element of *A*

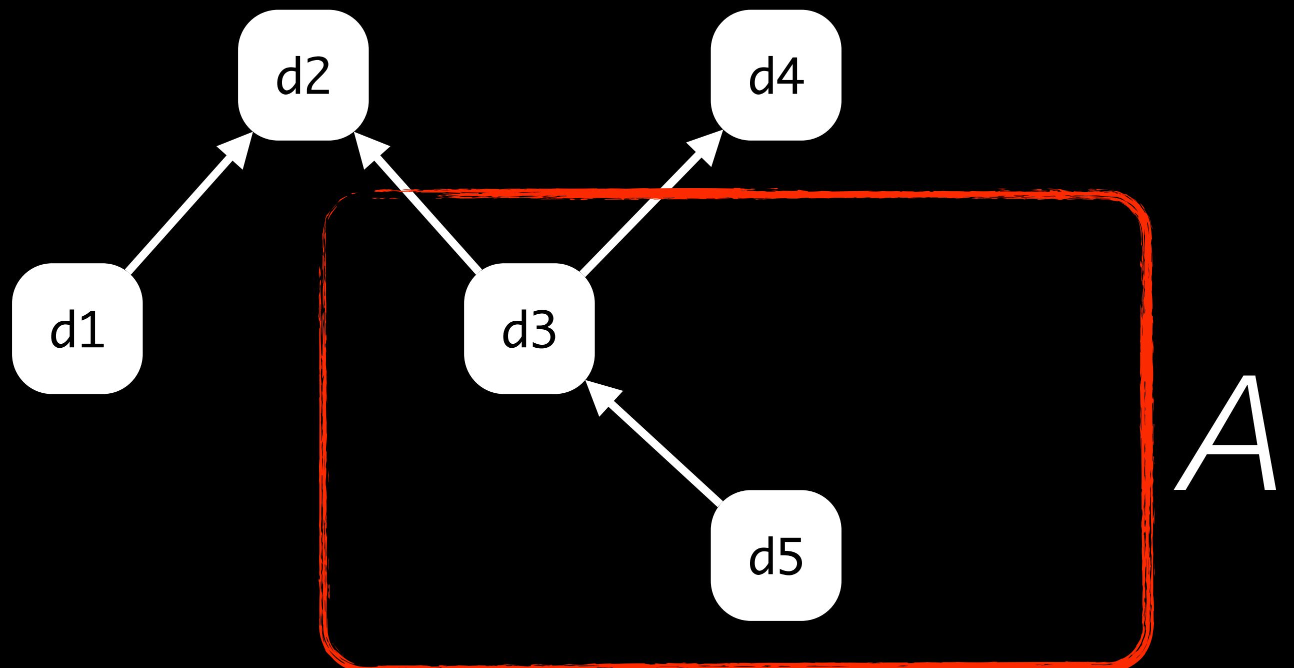
2. Analysis Abstraction
Least Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq U$, then z is a least upper bound of A if:



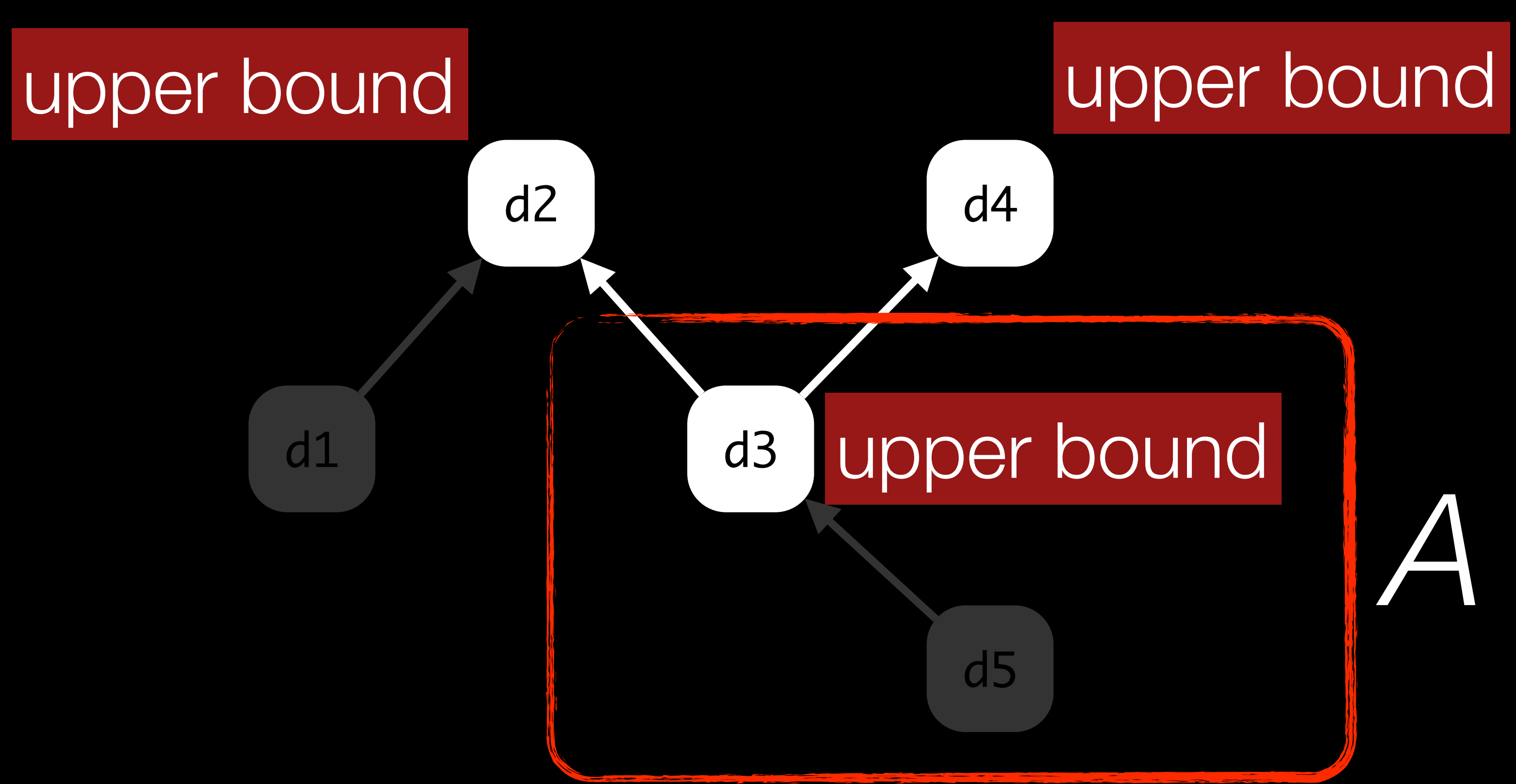
2. Analysis Abstraction
Least Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq$ U, then z is a least upper bound of A if:
  - $\forall\, x \in A : x \sqsubseteq z$



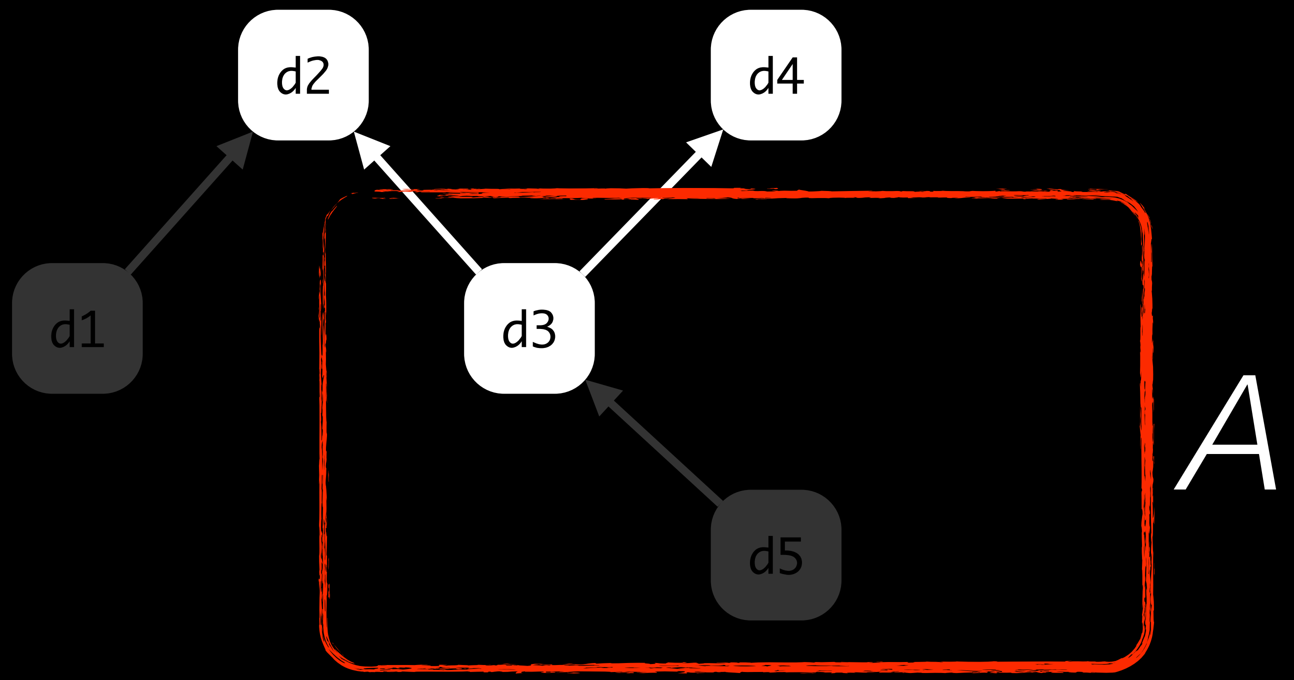
2. Analysis Abstraction
Least Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq U$, then z is a least upper bound of A if:
  - $\forall\, x \in A : x \sqsubseteq z$



2. Analysis Abstraction
Least Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq$ U, then z is a least upper bound of A if:
  - $\forall\, x \in A : x \sqsubseteq z$
  - $\forall\, y \in U : (\forall\, x \in A : x \sqsubseteq y) \implies z \sqsubseteq$ y

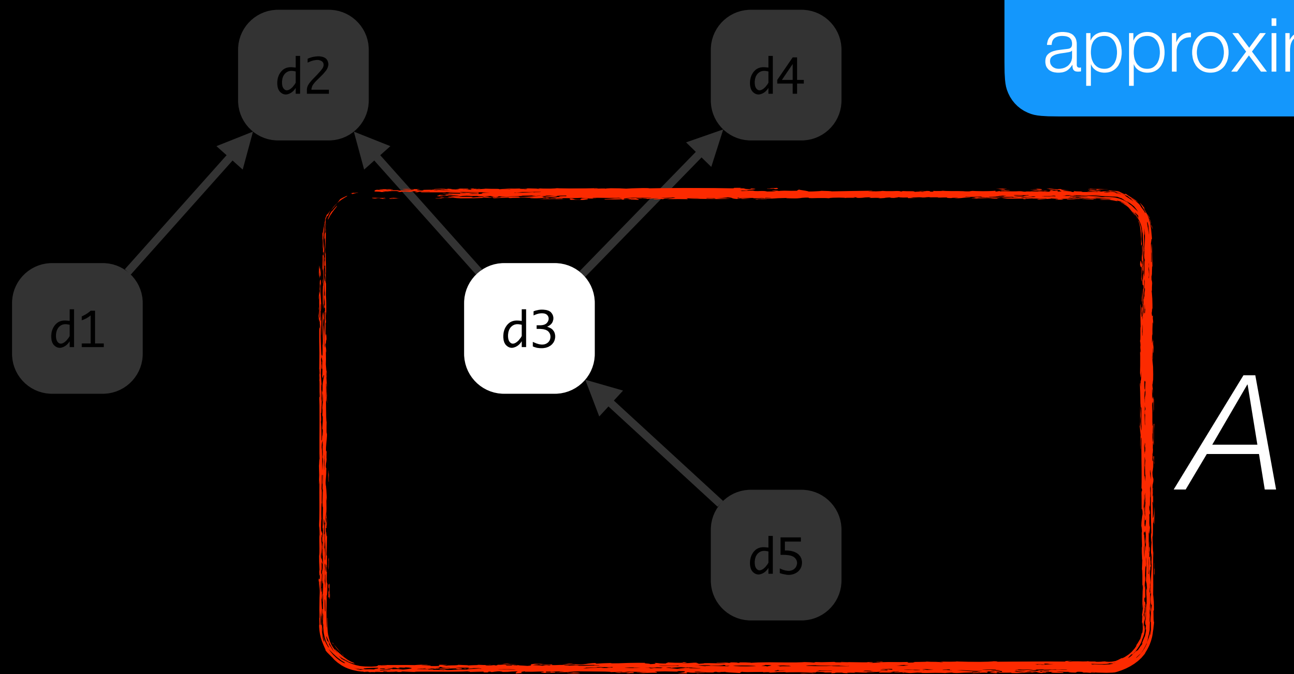


2. Analysis Abstraction
Least Upper Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq U$, then z is a least upper bound of A if:
  - $\forall\, x \in A : x \sqsubseteq z$
  - $\forall\, y \in U : (\forall\, x \in A : x \sqsubseteq y) \implies z \sqsubseteq y$

any upper bound y is an **approximation** of z, i.e., z is the **most precise** approximation of A

$z = \sqcup A$



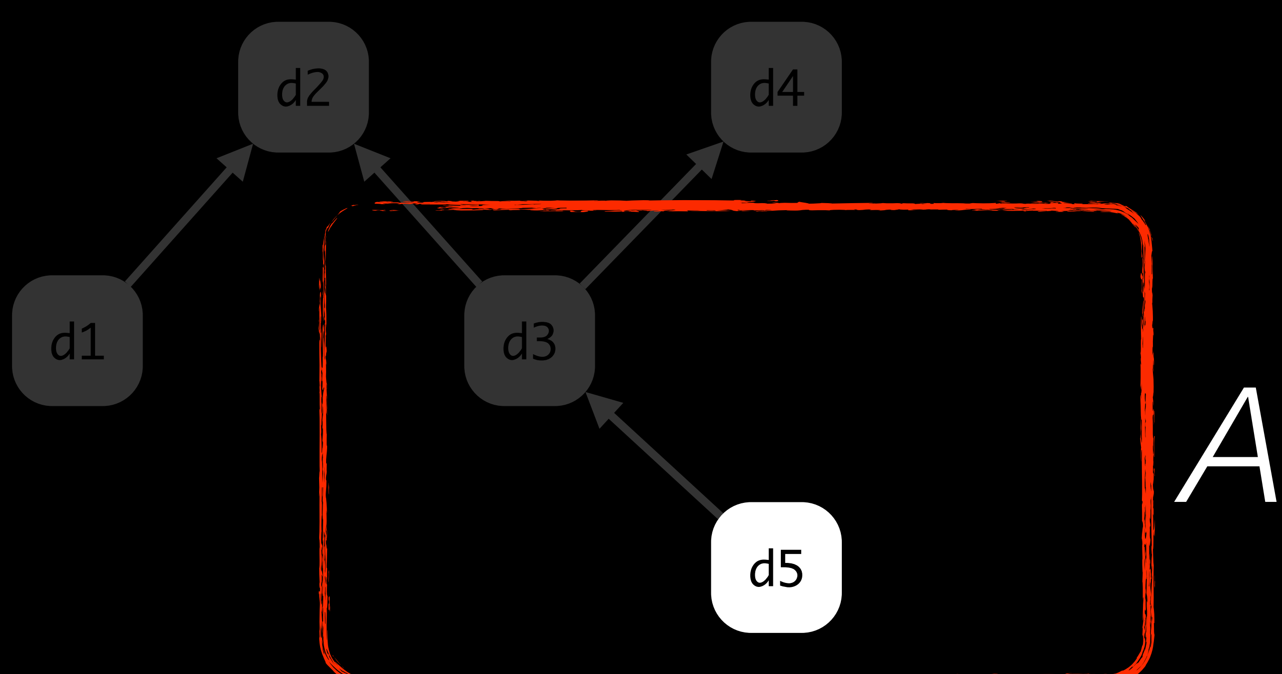
Question

- Can you have two least upper bounds?

2. Analysis Abstraction
Greatest Lower Bound

- If $(U, \sqsubseteq)$ is a poset and $A \subseteq U$, then z is a greatest lower bound of A if:
  - $\forall\, x \in A : z \sqsubseteq x$
  - $\forall\, y \in U : (\forall\, x \in A : y \sqsubseteq x) \implies y \sqsubseteq z$

$$z = \sqcap A$$

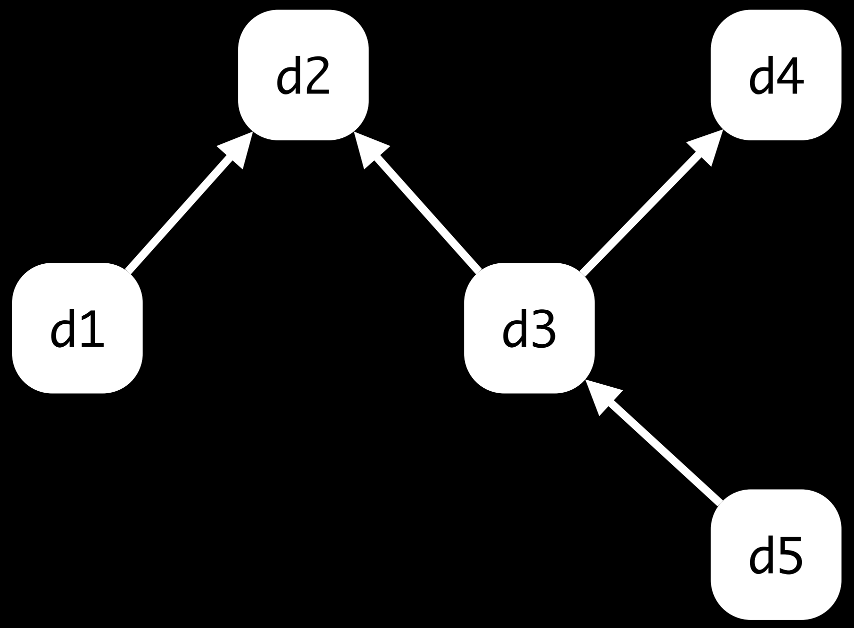


2. Analysis Abstraction
Lattice

- If (U, $\sqsubseteq$) is a poset where U $\neq \varnothing$, then (U, $\sqsubseteq$) is a lattice if $\forall\ x, y \in U$:
  - $\exists\ z \in U : z = x \sqcup y$ (Least Upper Bound)
  - $\exists\ z \in U : z = x \sqcap y$ (Greatest Lower Bound)

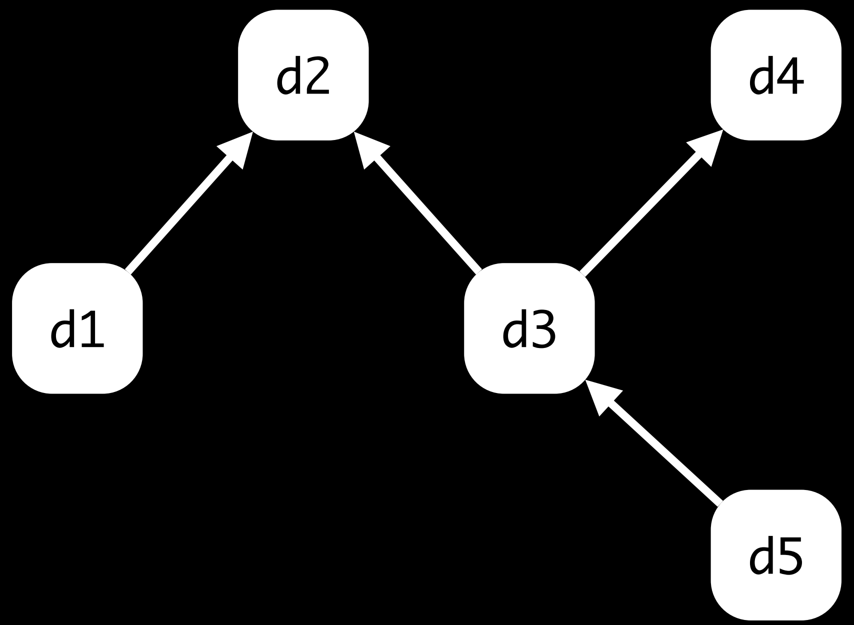
join

meet



2. Analysis Abstraction
Join Semi-Lattice

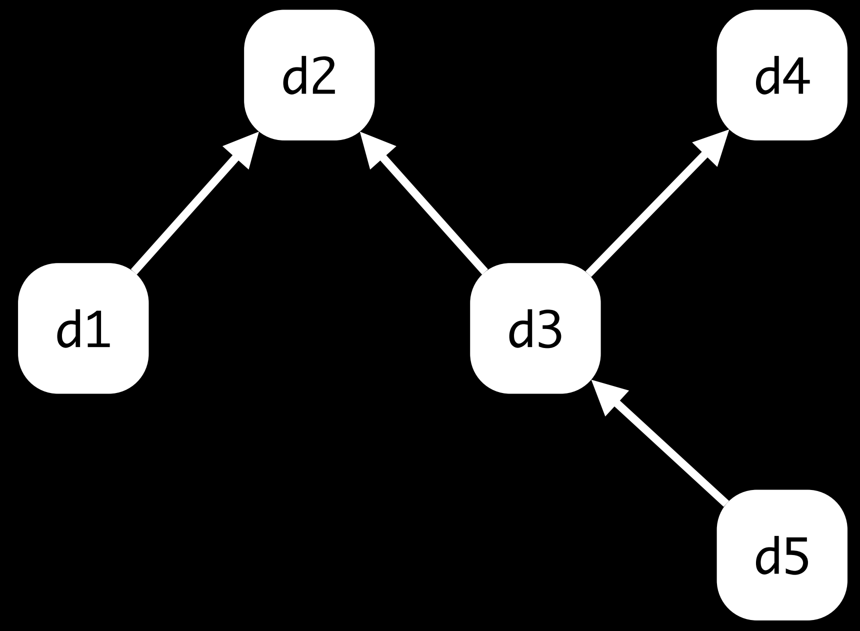
- If (U, $\sqsubseteq$) is a poset where U $\neq \varnothing$, then (U, $\sqsubseteq$) is a lattice if $\forall$ x, $y \in U$:
  - $\exists$ z $\in$ U : $z = x \sqcup y$ (Least Upper Bound) join



2. Analysis Abstraction
Meet Semi-Lattice

- If (U, $\sqsubseteq$) is a poset where U $\neq \varnothing$, then (U, $\sqsubseteq$) is a lattice if $\forall$ x, $y \in U$:
  - $\exists$ z $\in U$: $z = x \sqcap y$ (Greatest Lower Bound)

meet

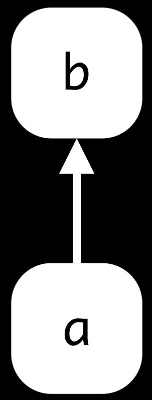


2. Analysis Abstraction
Is that a lattice?

a

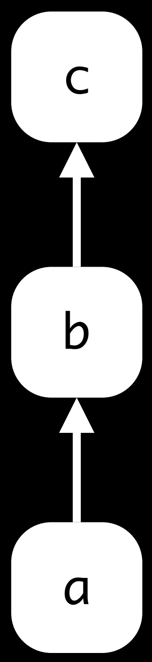
yes

2. Analysis Abstraction
Is that a lattice?



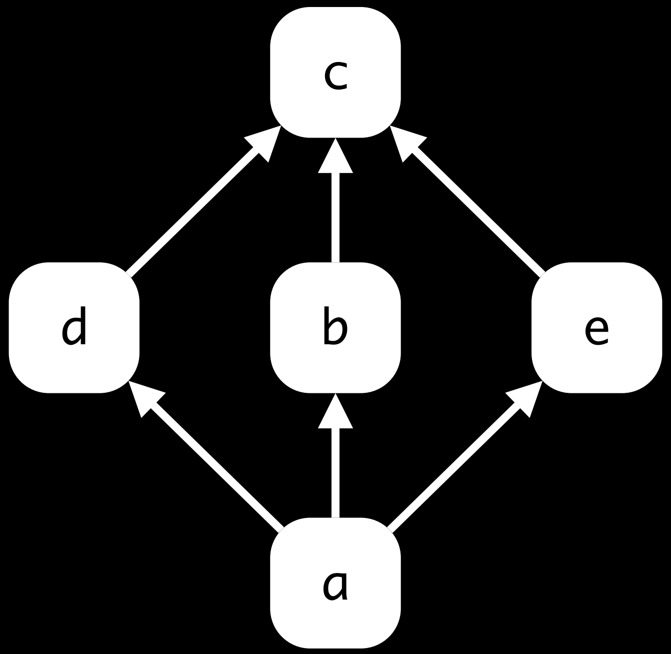
yes

2. Analysis Abstraction
Is that a lattice?



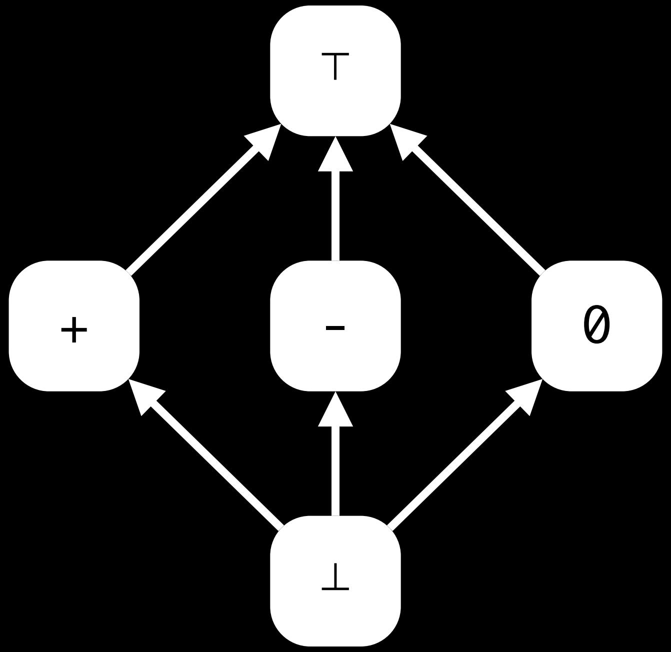
yes

2. Analysis Abstraction
Is that a lattice?



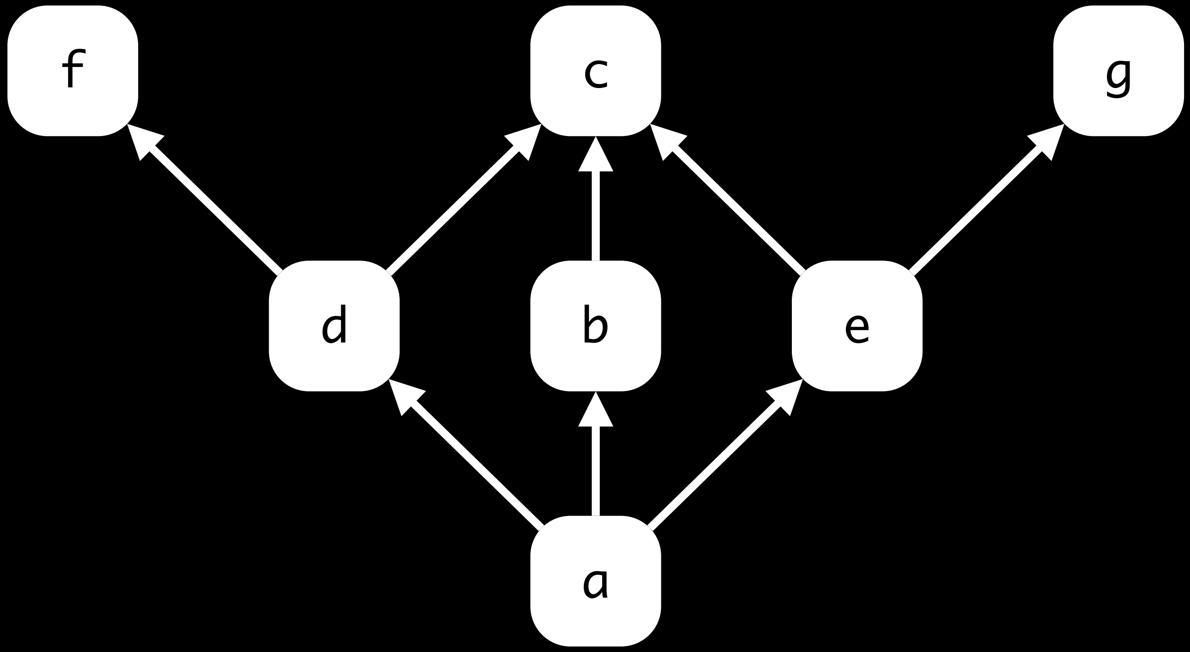
yes

2. Analysis Abstraction
Is that a lattice?



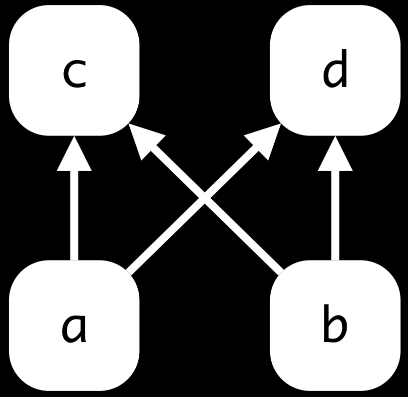
Sign Lattice

2. Analysis Abstraction
Is that a lattice?



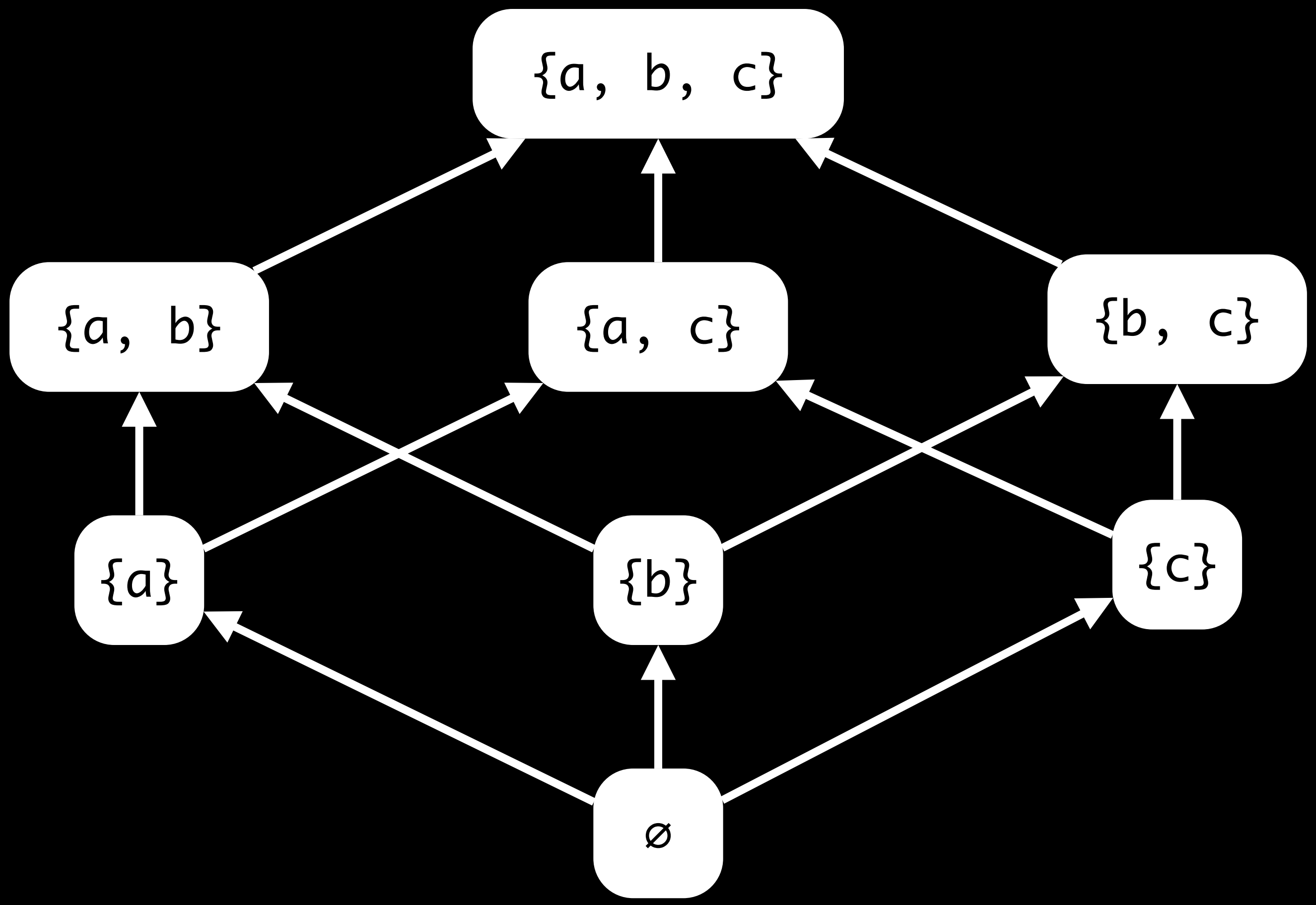
no

2. Analysis Abstraction
Is that a lattice?



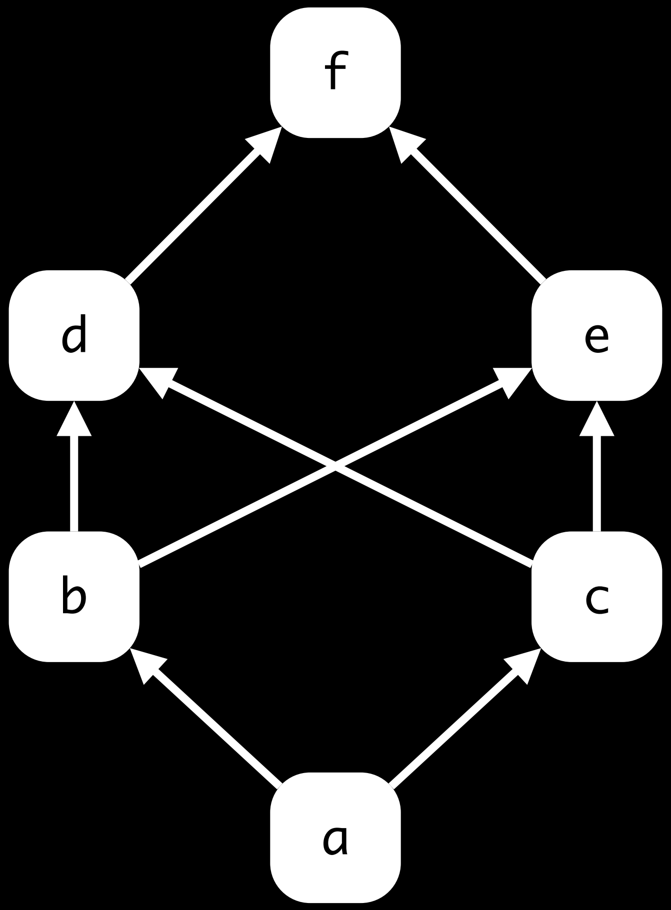
no

2. Analysis Abstraction
Is that a lattice?



yes

2. Analysis Abstraction
Is that a lattice?



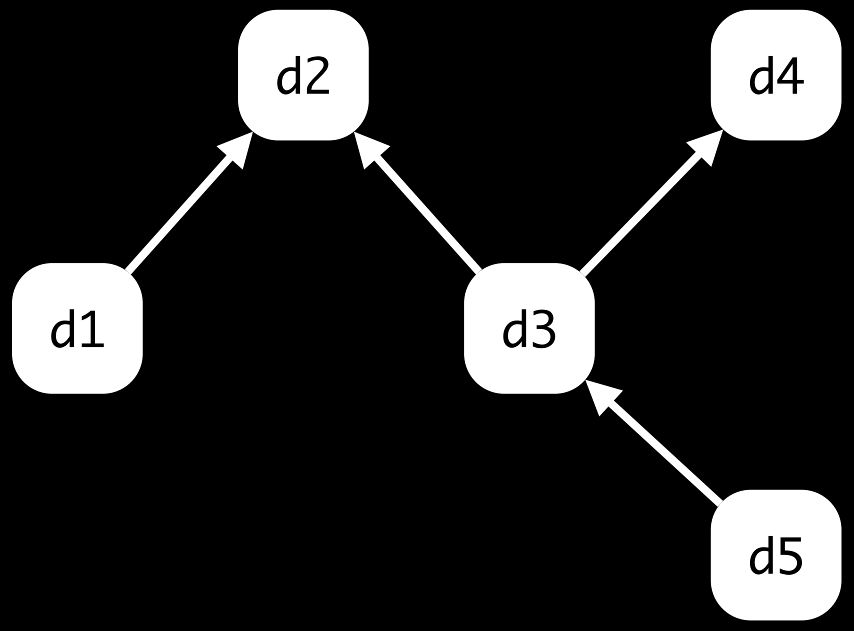
no

Key Takeaway About Lattices

- A lattice is a partial order (U, $\sqsubseteq$) that comes equipped with
  - A binary operator join, $\sqcup$, which computes the least upper bound
  - A binary operator meet, $\sqcap$, which computes the greatest lower bound

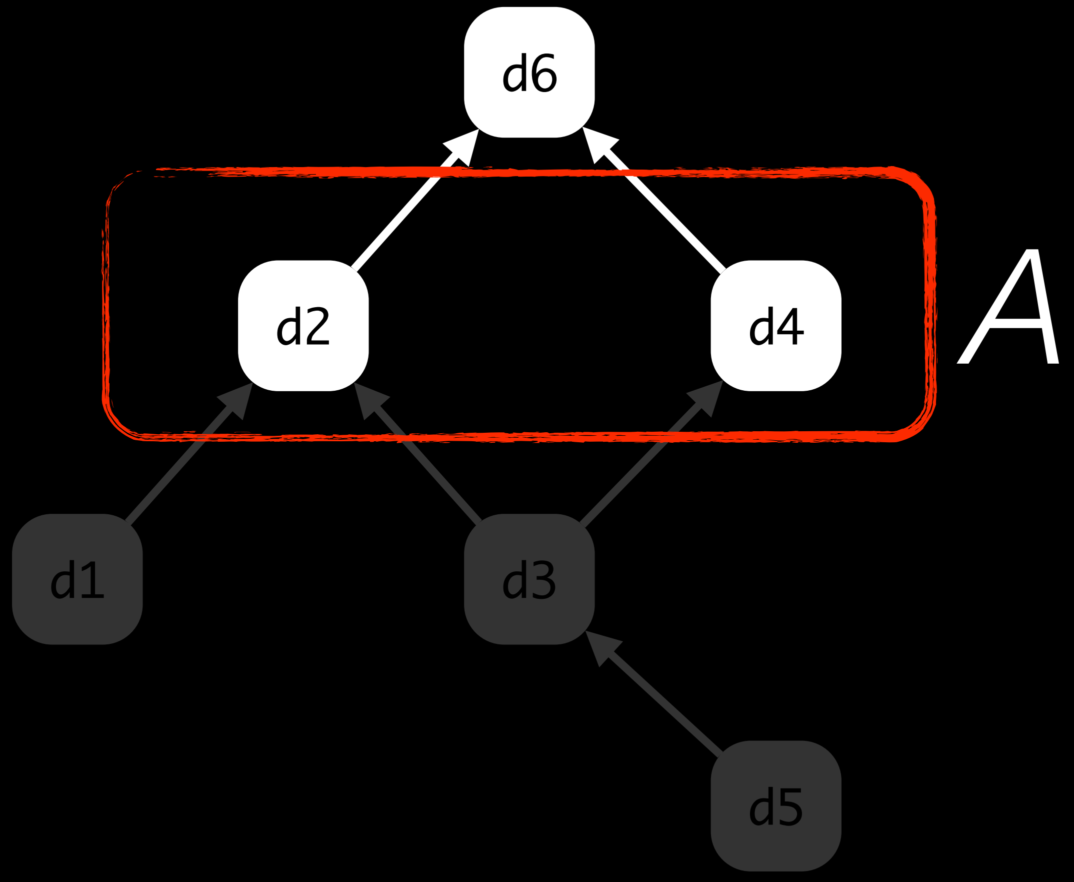
2. Analysis Abstraction
Complete Lattice

- If (U, $\sqsubseteq$) is a poset where $U \neq \varnothing$, then (U, $\sqsubseteq$) is a complete lattice if $\forall A \subseteq U$:
  - $\exists z \in U : z = \sqcup A$ (Least Upper Bound)



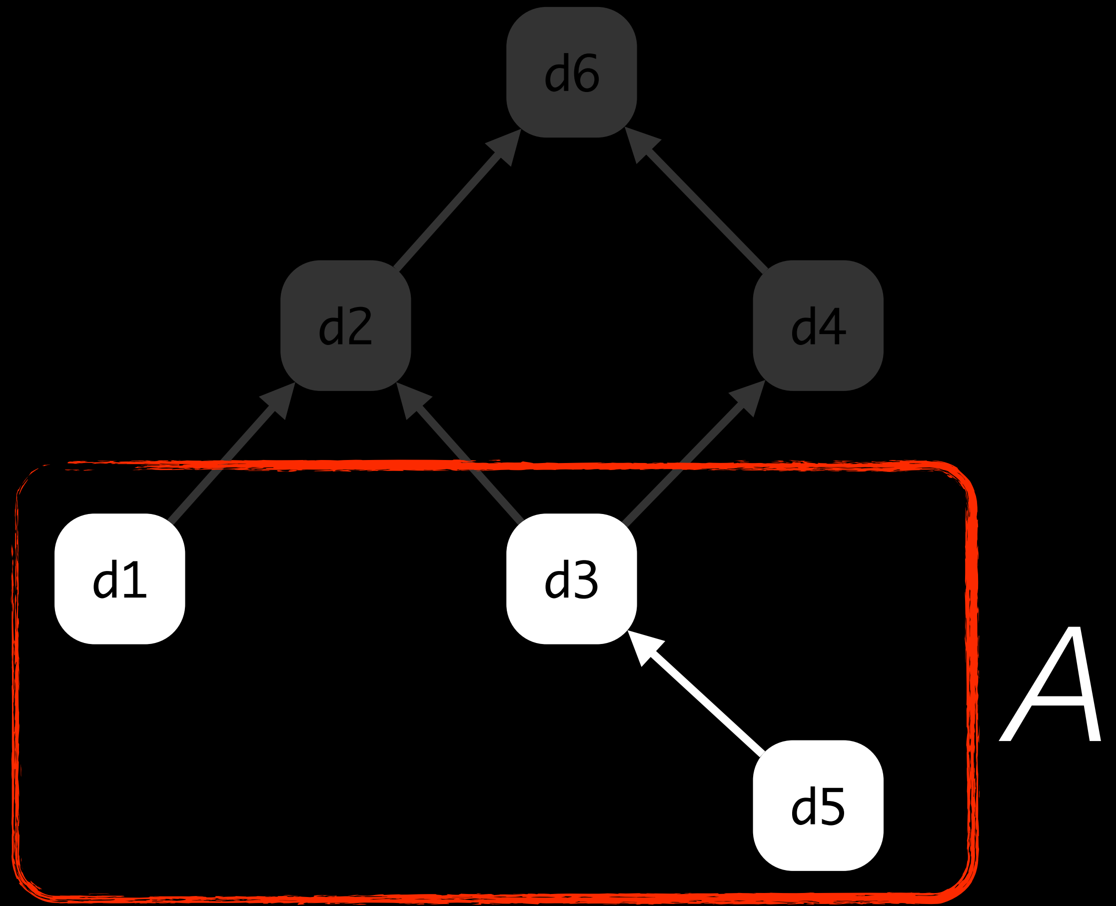
2. Analysis Abstraction
Complete Lattice

- If $(U, \sqsubseteq)$ is a poset where $U \neq \varnothing$, then $(U, \sqsubseteq)$ is a complete lattice if $\forall A \subseteq U$:
  - $\exists z \in U : z = \sqcup A$ (Least Upper Bound)



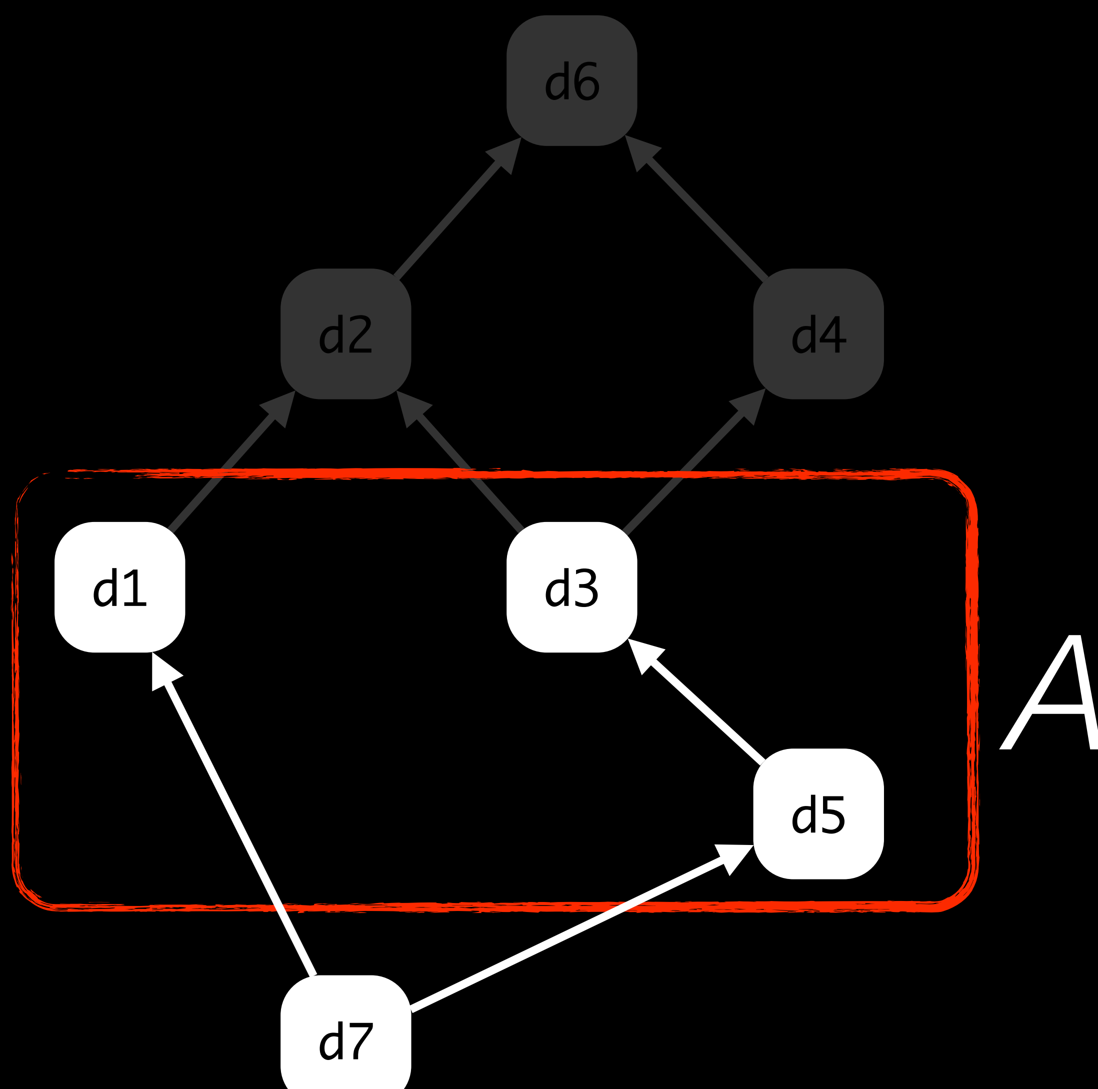
2. Analysis Abstraction
Complete Lattice

- If (U, $\sqsubseteq$) is a poset where $U \neq \emptyset$, then (U, $\sqsubseteq$) is a complete lattice if $\forall A \subseteq U$:
  - $\exists z \in U : z = \sqcup A$ (Least Upper Bound)
  - $\exists z \in U : z = \sqcap A$ (Greatest Lower Bound)



2. Analysis Abstraction
Complete Lattice

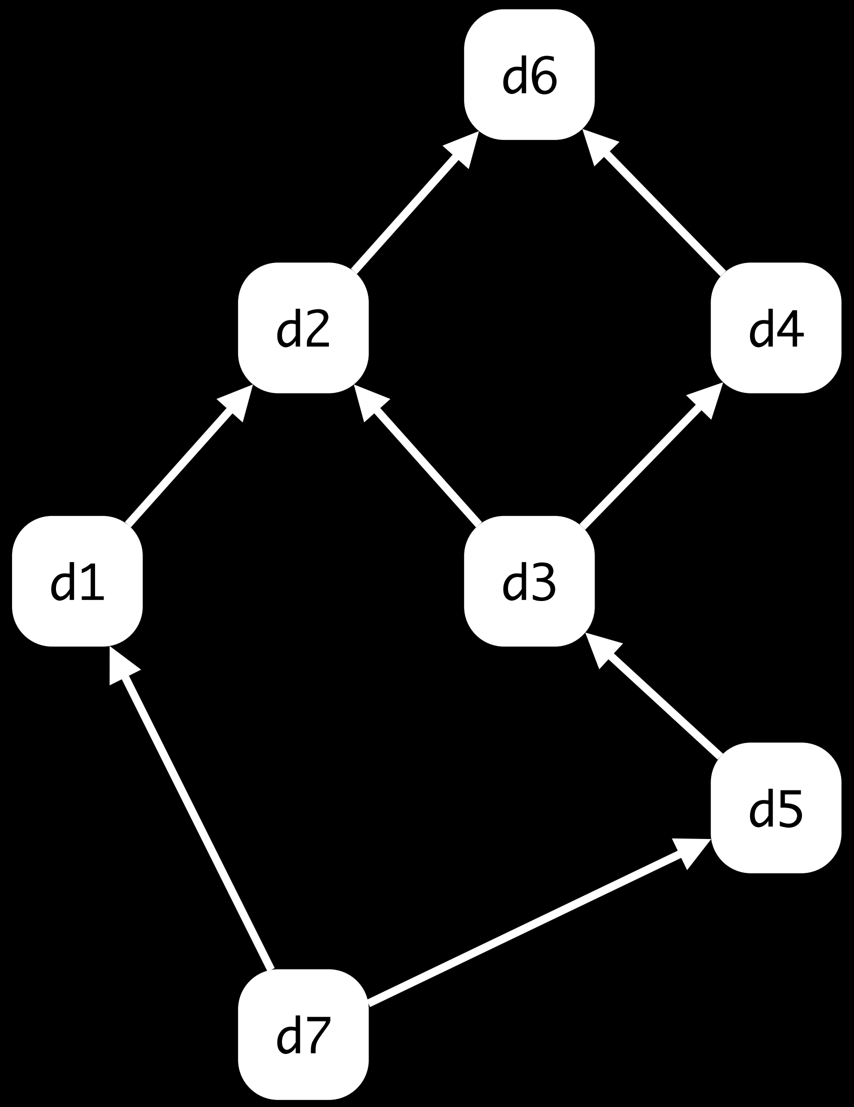
- If $(U, \sqsubseteq)$ is a poset where $U \neq \varnothing$, then $(U, \sqsubseteq)$ is a complete lattice if $\forall A \subseteq U$:
  - $\exists z \in U : z = \sqcup A$ (Least Upper Bound)
  - $\exists z \in U : z = \sqcap A$ (Greatest Lower Bound)



2. Analysis Abstraction
Complete Lattice

- If $(U, \sqsubseteq)$ is a poset where $U \neq \varnothing$, then $(U, \sqsubseteq)$ is a complete lattice if $\forall\, A \subseteq U$:
  - $\exists\, z \in U : z = \sqcup A$ (Least Upper Bound)
  - $\exists\, z \in U : z = \sqcap A$ (Greatest Lower Bound)

$(U, \sqsubseteq)$ is a complete lattice

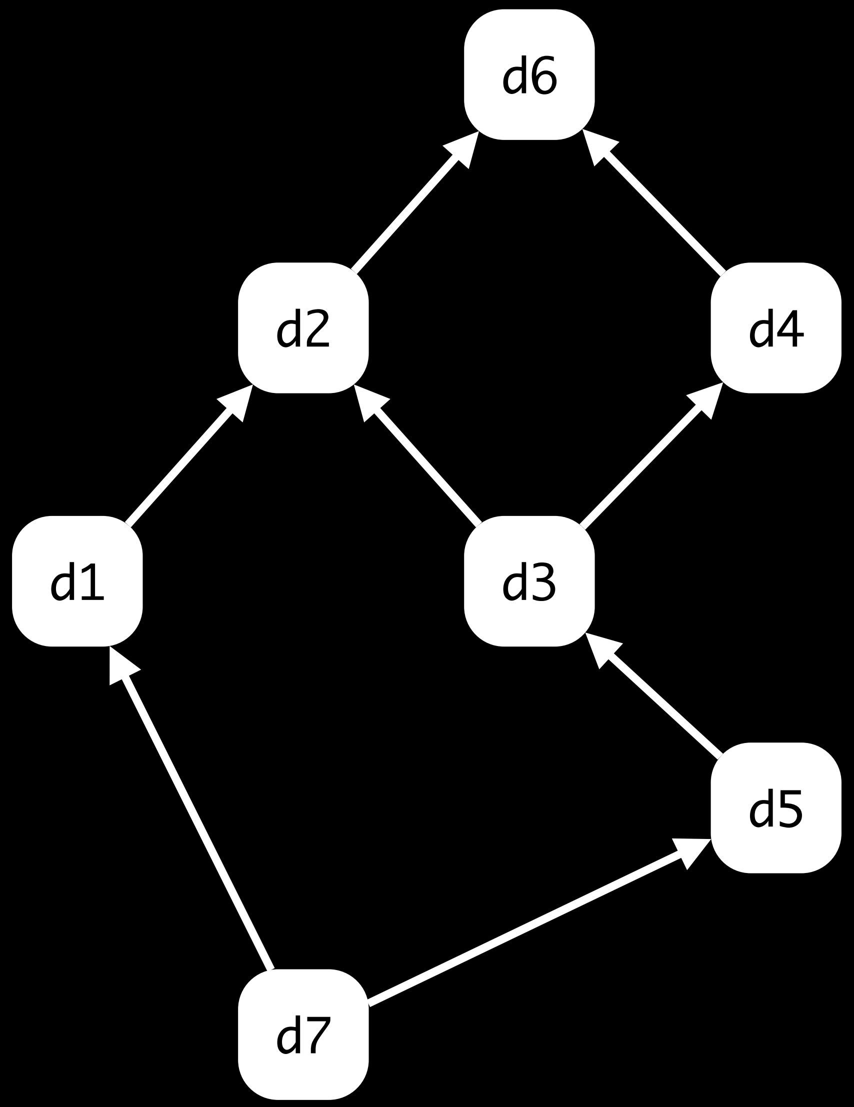


Key Takeaway About Complete Lattices

- A complete lattice is a lattice where the meet and join operators extend to arbitrary subsets of the lattice.

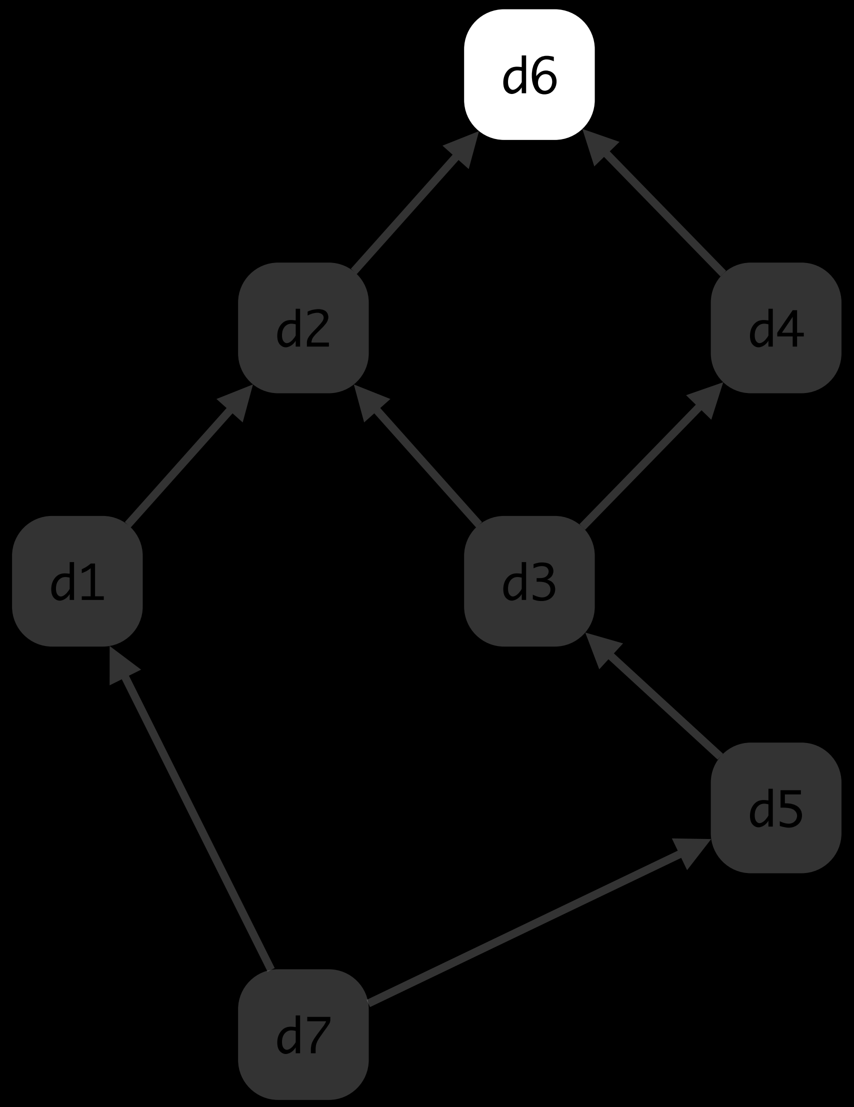
2. Analysis Abstraction
Bounded Lattice

- If (U, $\sqsubseteq$) is a lattice, then (U, $\sqsubseteq$) is bounded if:



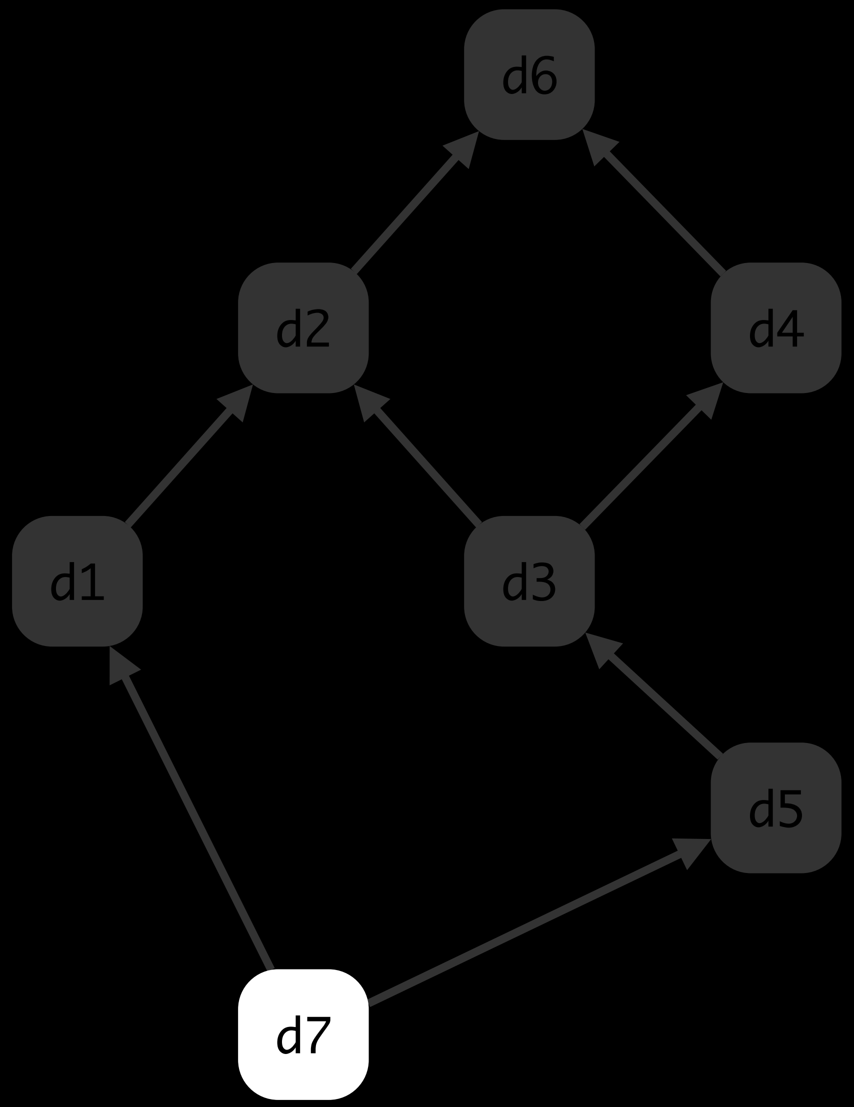
2. Analysis Abstraction
Bounded Lattice

- If $(U, \sqsubseteq)$ is a lattice, then $(U, \sqsubseteq)$ is bounded if:
  - $\exists\ z \in U : (\forall\ x \in U : x \sqsubseteq z) = \sqcup\ U$ (Top or $\top$)



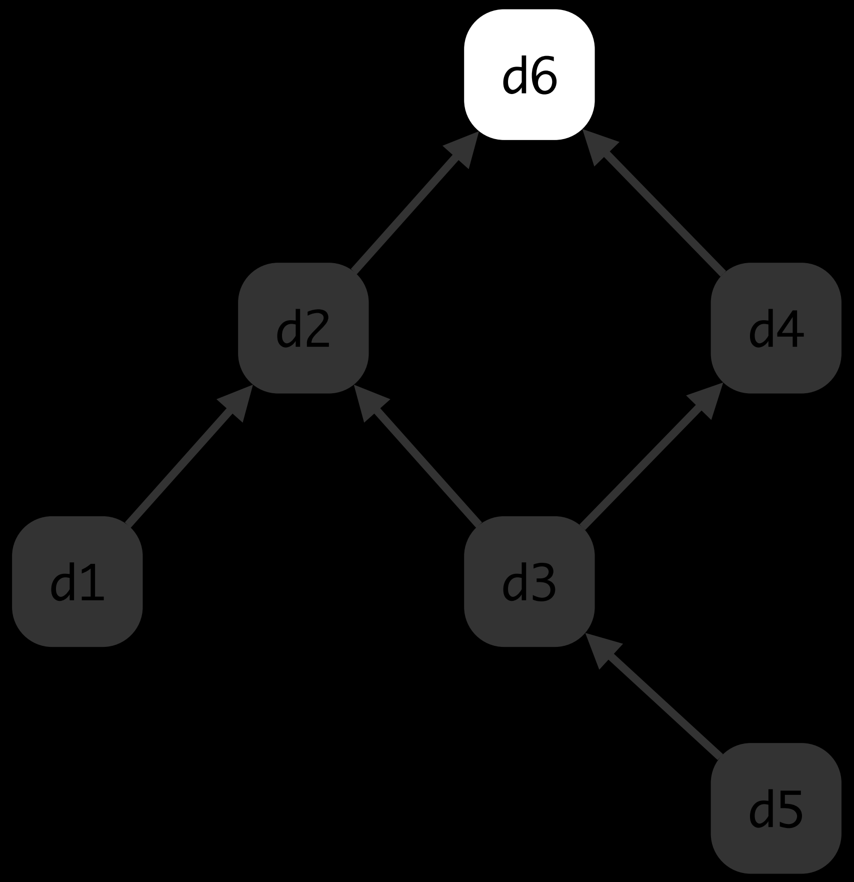
2. Analysis Abstraction
Bounded Lattice

- If $(U, \sqsubseteq)$ is a lattice, then $(U, \sqsubseteq)$ is bounded if:
    - $\exists\ z \in U : (\forall\ x \in U : x \sqsubseteq z) = \sqcup\ U$ (Top or $\top$)
    - $\exists\ z \in U : z = \sqcap\ U$ (Bottom or $\perp$)



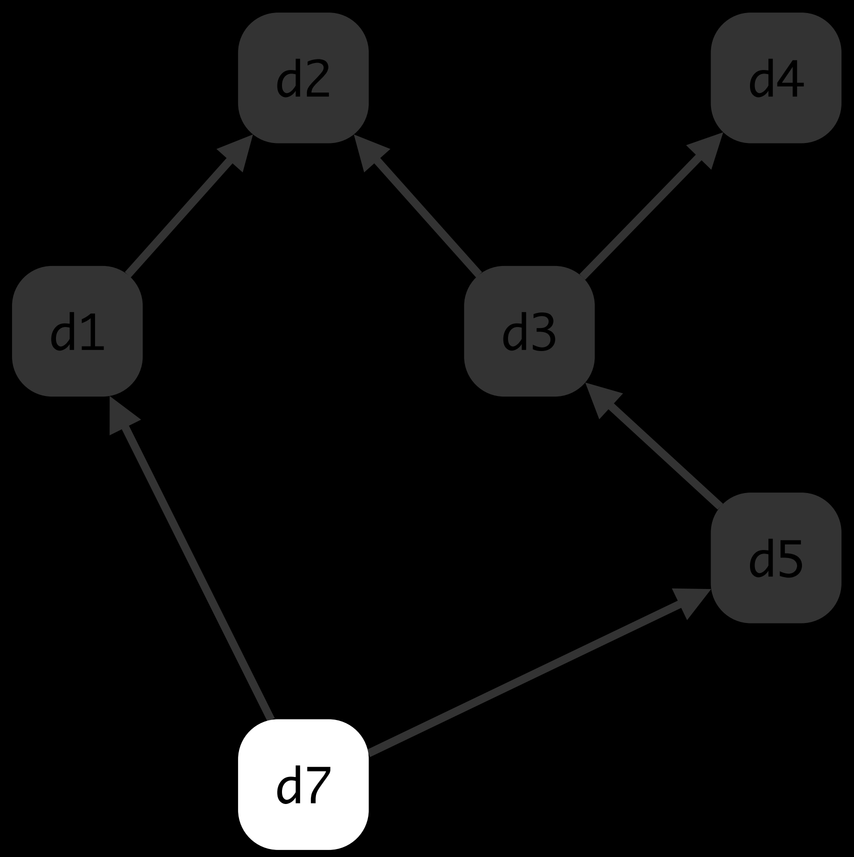
2. Analysis Abstraction
Bounded Join Semi-Lattice

- If (U, ⊑) is a join semi-lattice, then (U, ⊑) is a bounded join semi-lattice if:
  - $\exists\ z \in U : z = \sqcup\ U$ (Top or ⊤)



2. Analysis Abstraction
Bounded Meet Semi-Lattice

- If $(U, \sqsubseteq)$ is a meet semi-lattice, then $(U, \sqsubseteq)$ is a bounded meet semi-lattice if:
  - $\exists\ z \in U : z = \sqcap\, U$ (Bottom or $\bot$)



2. Analysis Abstraction
Lattice Questions

- **Q**: Is a complete lattice bounded?
- **Q**: Is a finite lattice complete?

2. Analysis Abstraction
Lattice Questions

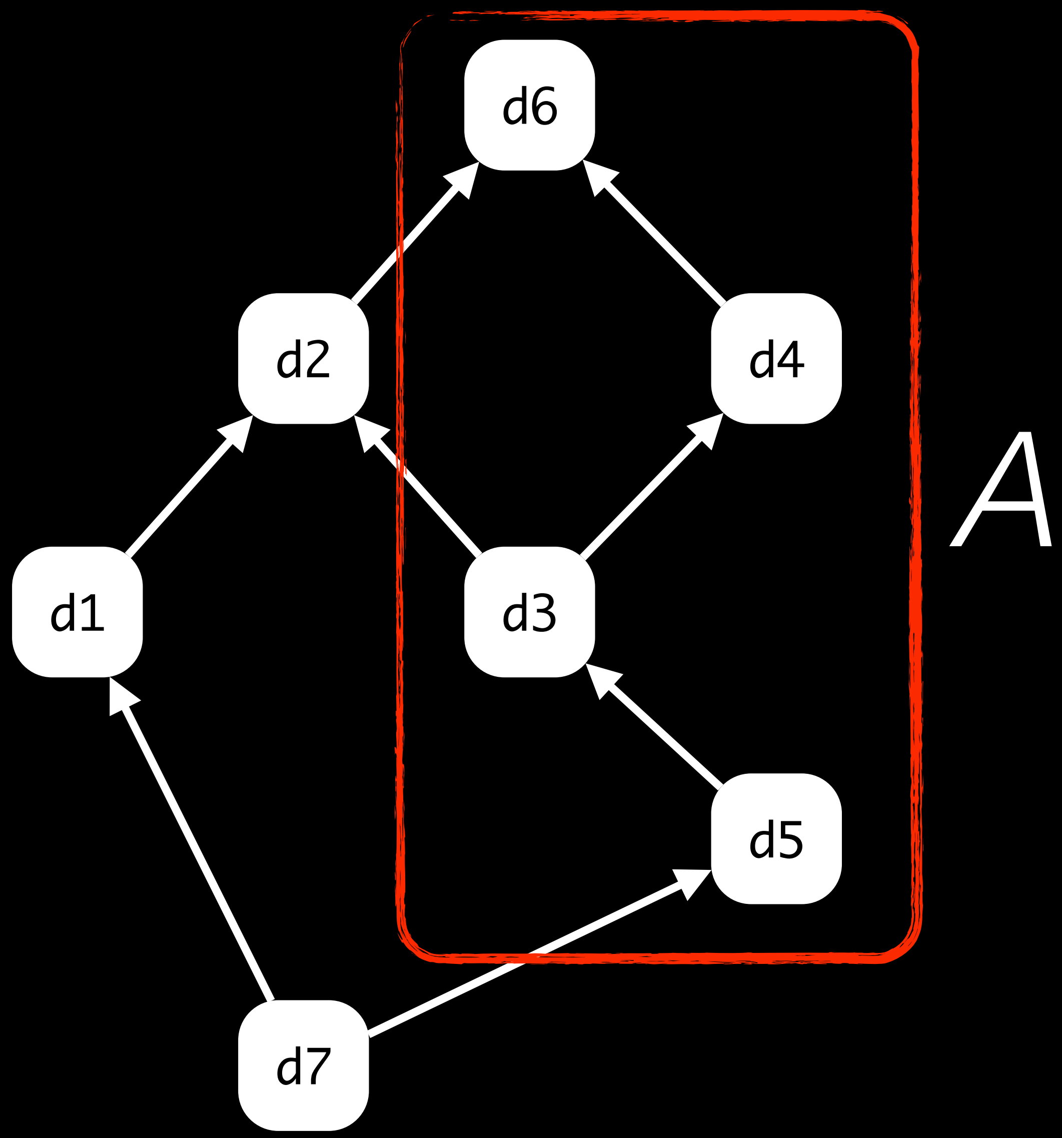
- Let $U = \{ F \mid F \subseteq \mathbb{N}, F \text{ finite} \} \cup \{ \mathbb{N} \setminus F \mid F \subseteq \mathbb{N}, F \text{ finite} \}$
- **Q:** Is $(U, \subseteq)$ a lattice?
- **Q:** Is it bounded?
- **Q**: Is it complete?

Key Takeaway About Bounded Lattices

- A bounded lattice comes with a top element and a bottom element

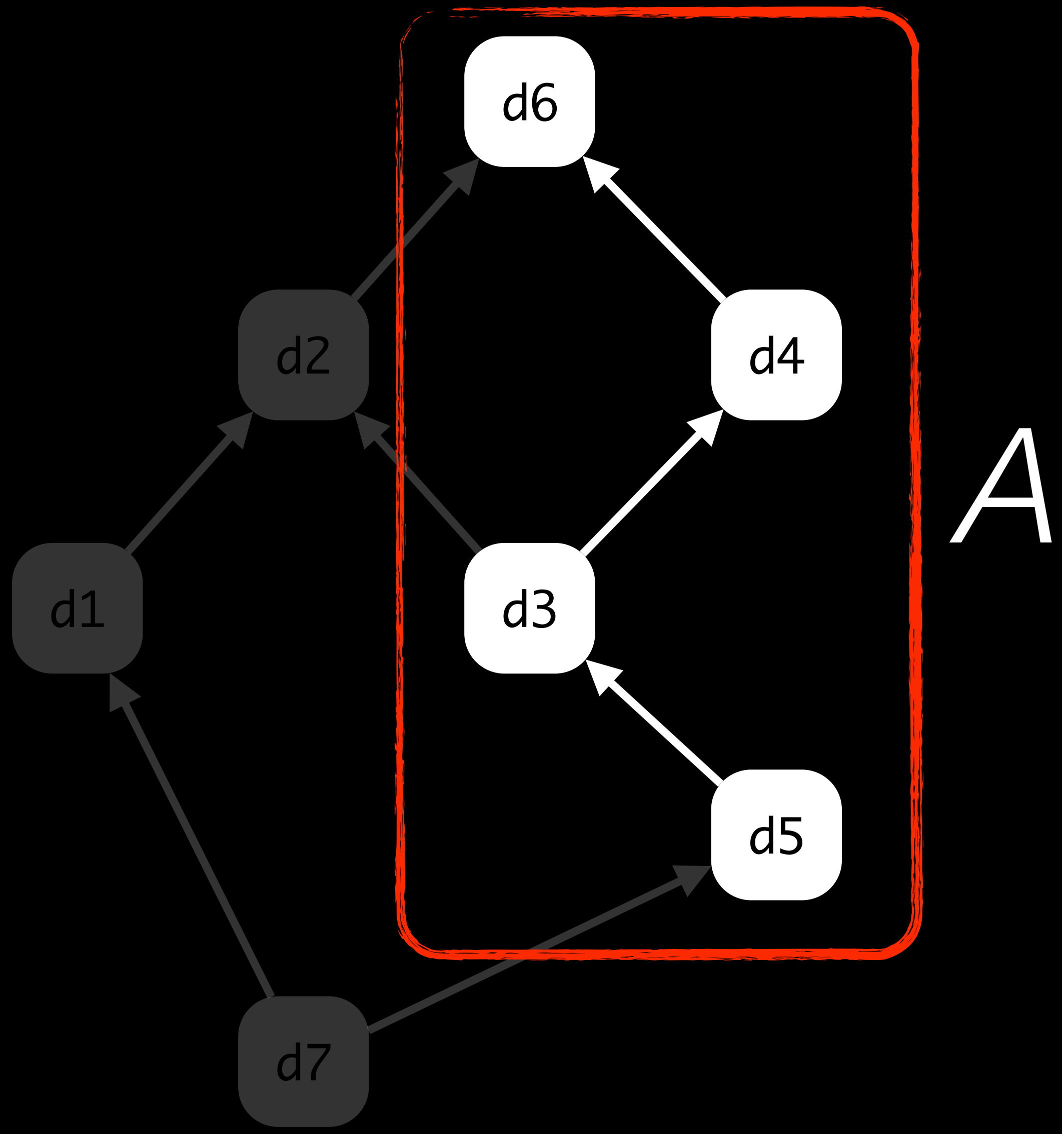
2. Analysis Abstraction
Lattice Chain

- If $(U, \sqsubseteq)$ is a lattice, then $A \subseteq U$ is a chain if:



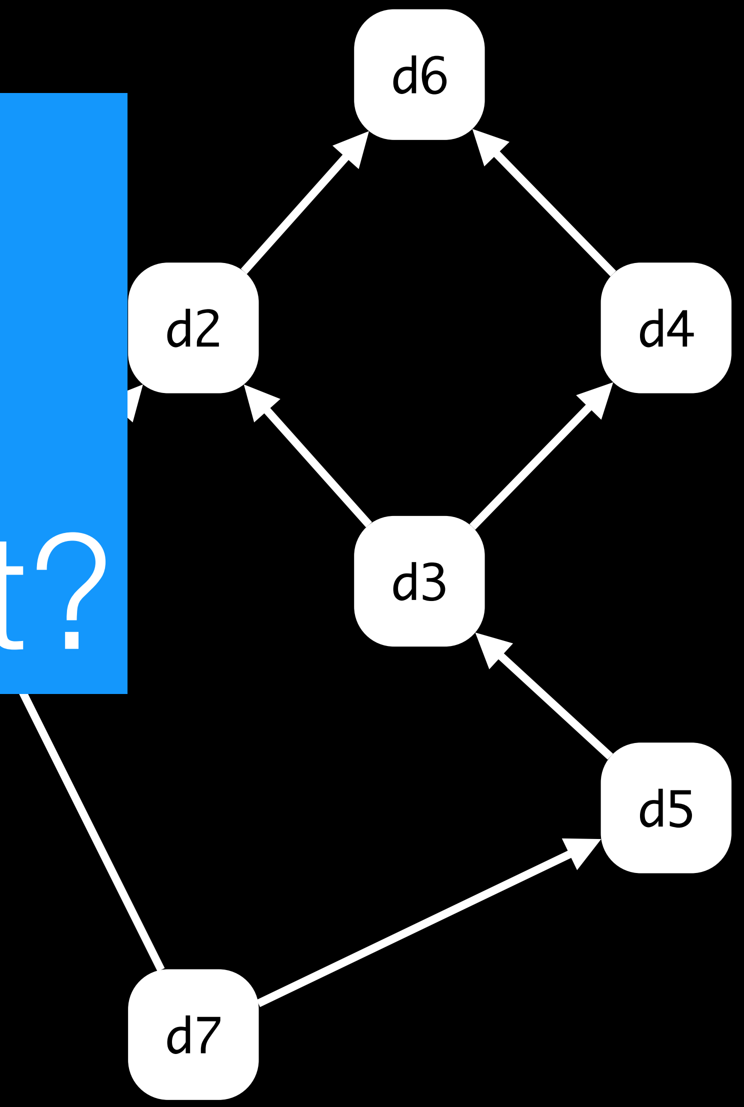
2. Analysis Abstraction
Lattice Chain

- If $(U, \sqsubseteq)$ is a lattice, then $A \subseteq U$ is a chain if:
  - $\forall\, x, y \in A : x \sqsubseteq y \vee y \sqsubseteq x$



2. Analysis Abstraction
Lattice Height

- If (U, $\sqsubseteq$) is a lattice, then the lattice height is the cardinality of the **longest** chain in the lattice

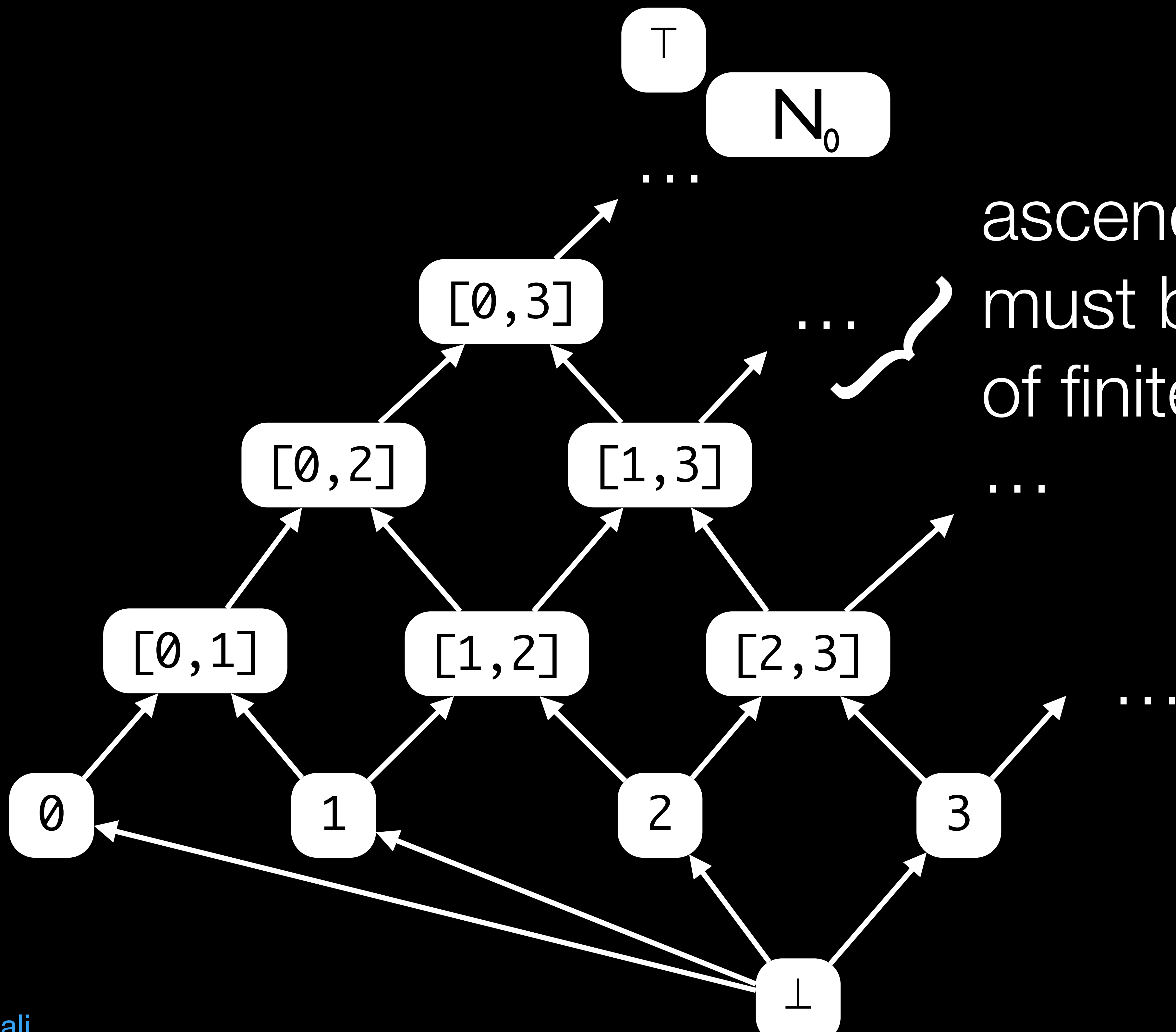


Does the lattice have a finite height?

2. Analysis Abstraction
Ascending Chain Condition

- A lattice satisfies the ascending chain condition if, for any sequence $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$,
  - $\exists\, m \in \mathbb{N} : \forall\, n \geq m,\ d_n = d_m$

2. Analysis Abstraction
Ascending Chain Condition



ascending chains must be of finite length

2. Analysis Abstraction
Ascending Chain Condition

- Lattice may be infinite as long as every ascending chain eventually stabilizes
  - $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$, in other words
  - $\exists\, n \in \mathbb{N} : d_n = d_{n+1}$

2. Analysis Abstraction
Ascending Chains Questions

- **Q**: Can a finite lattice have an ascending chain that does not stabilize?

Key Takeaways About Lattice Chains

- A chain is a totally ordered subset of the lattice
- The height of the lattice is the size of the largest chain

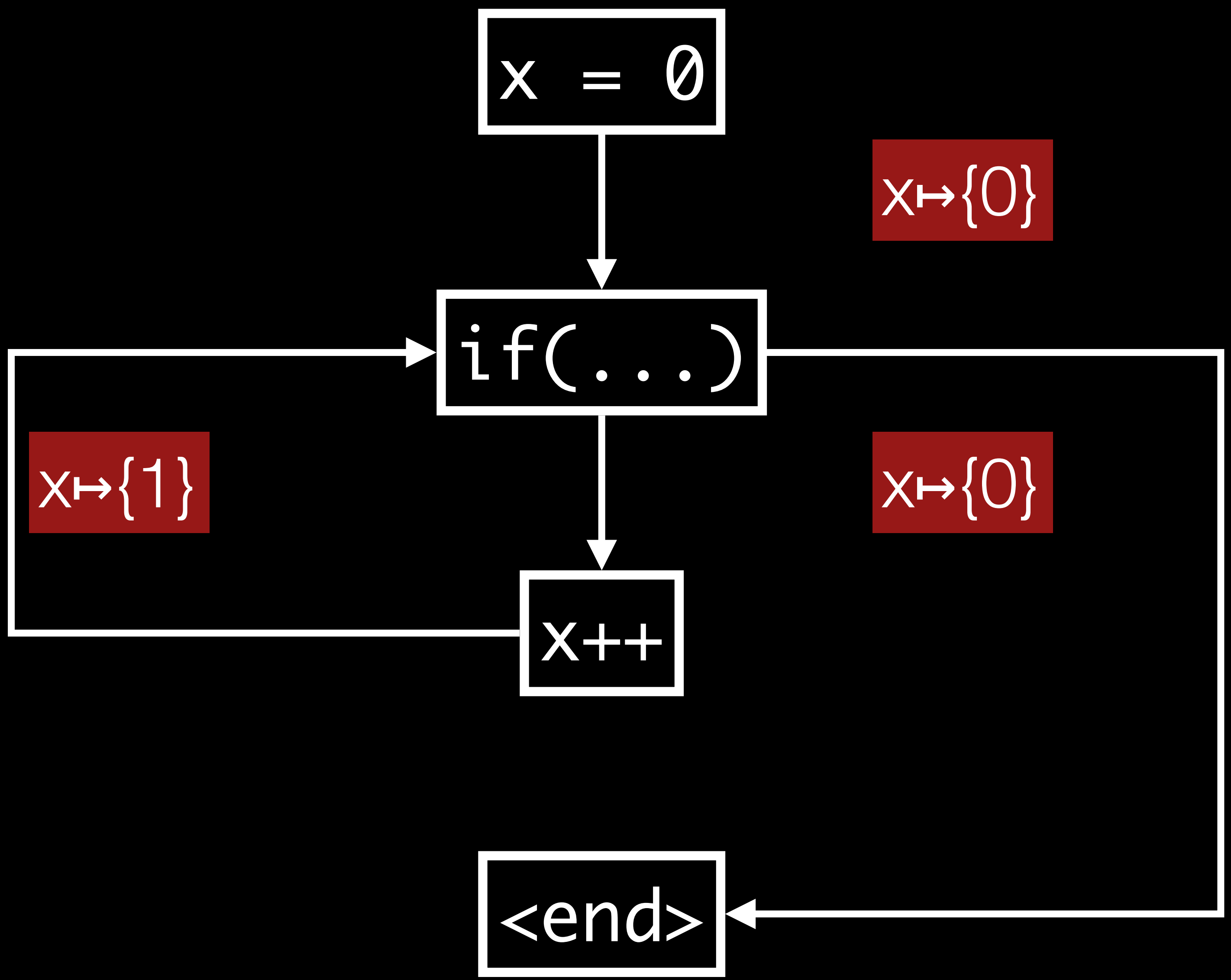
2. Analysis Abstraction
Types of Lattices

- **Powerset Lattice**: if F is a set, then the powerset $\mathcal{P}(F)$ with $\sqsubseteq$ defined as $\subseteq$ (or as $\supseteq$) is a lattice.
- **Product Lattice**: if L_A and L_B are lattices, then their product $L_A \times L_B$ with $\sqsubseteq$ defined as $(a_1, b_1) \sqsubseteq (a_2, b_2)$ if $a_1 \sqsubseteq a_2$ and $b_1 \sqsubseteq b_2$ is also a lattice.
- **Map Lattice**: if F is a set and L is a lattice, then the set of maps $F \rightarrow L$ with $\sqsubseteq$ defined as $m_1 \sqsubseteq m_2$ if $\forall_{f \in F}\, m_1(f) \sqsubseteq m_2(f)$ is also a lattice.

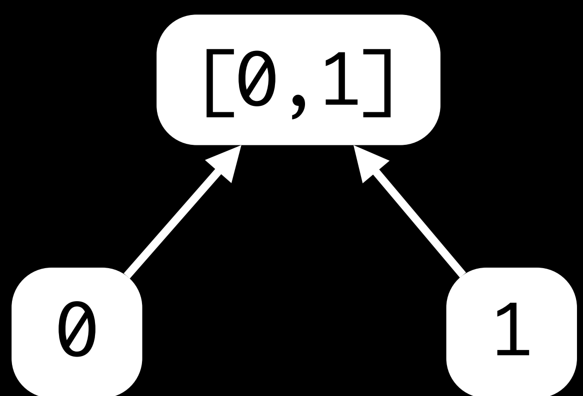
2. Analysis Abstraction

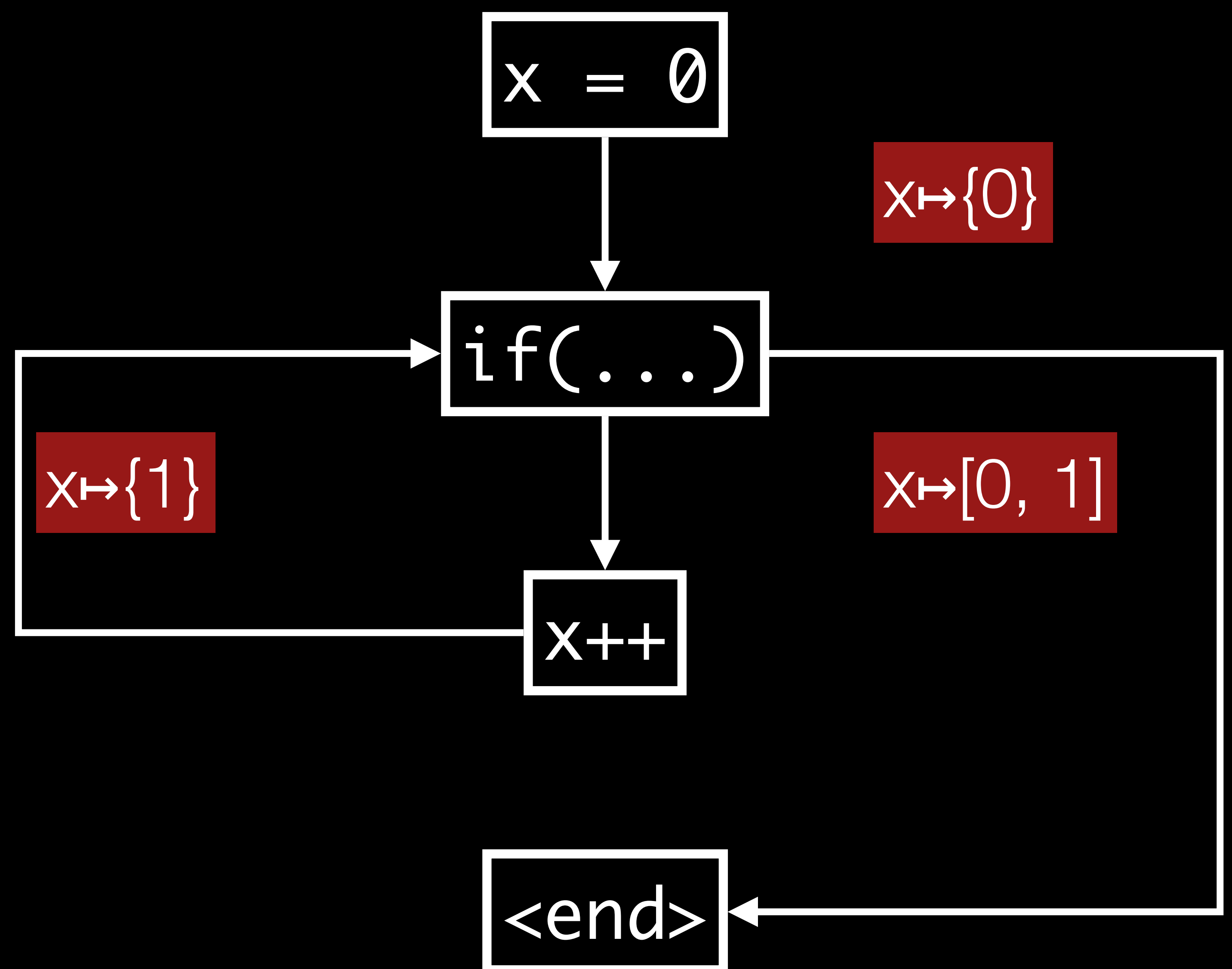
… so let’s finally use a lattice!

2. Analysis Abstraction
Lattice Example

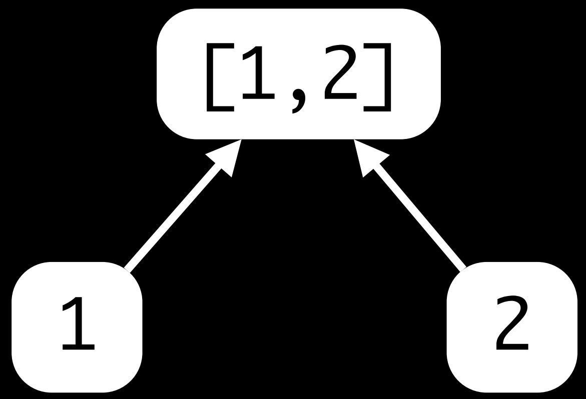


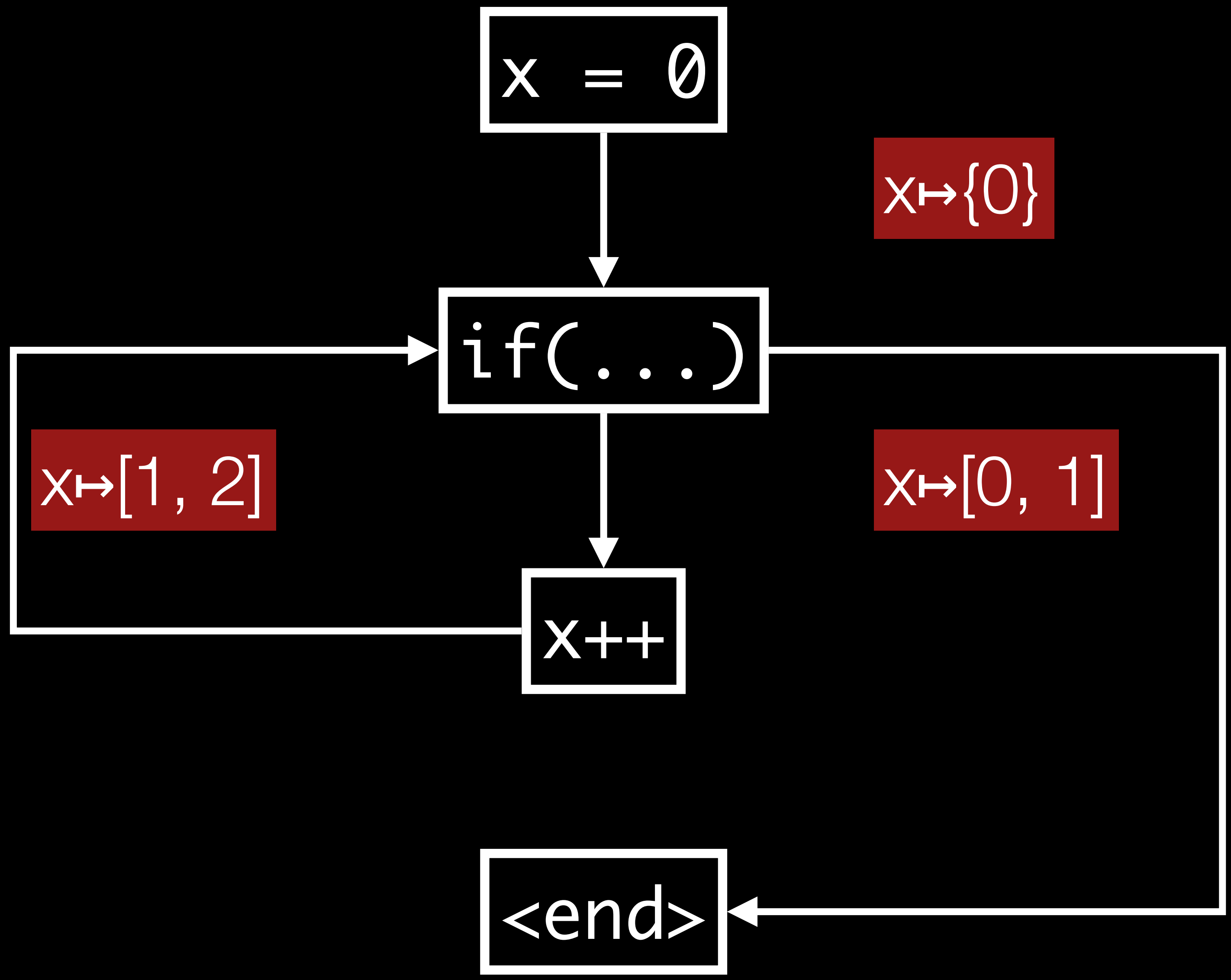
2. Analysis Abstraction
Lattice Example



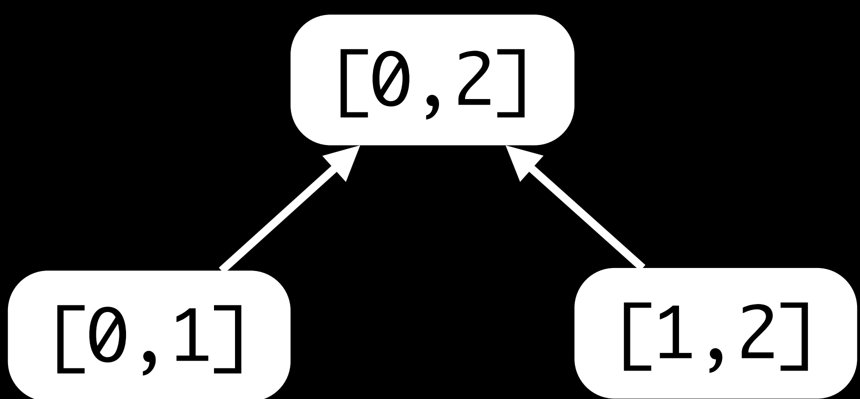


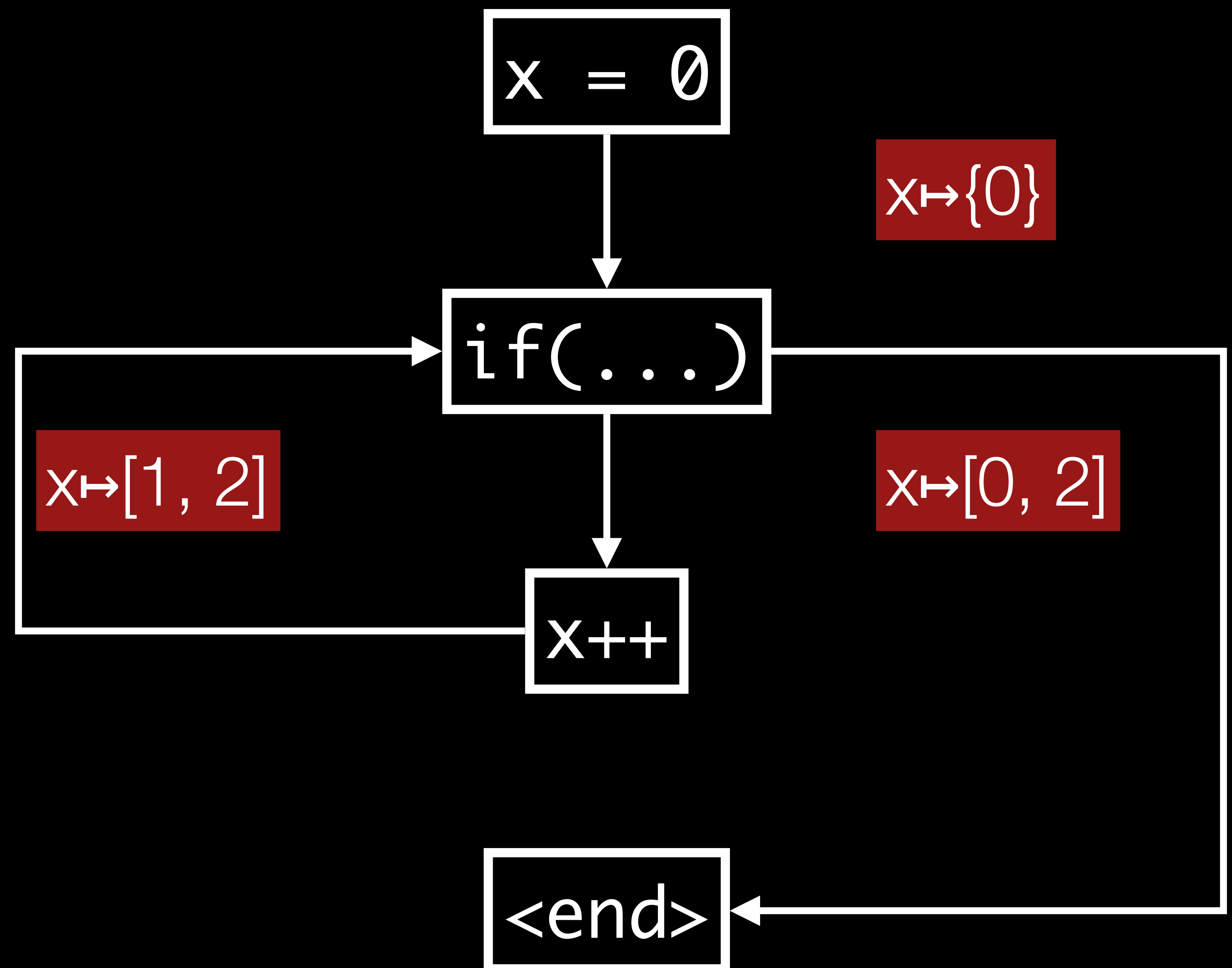
2. Analysis Abstraction
Lattice Example



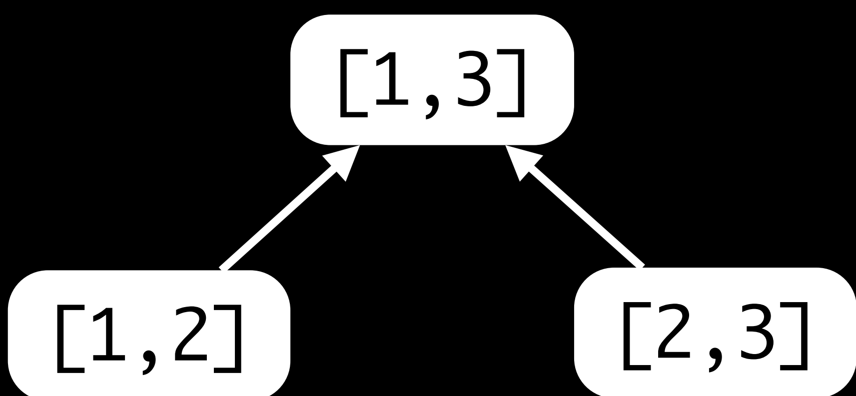


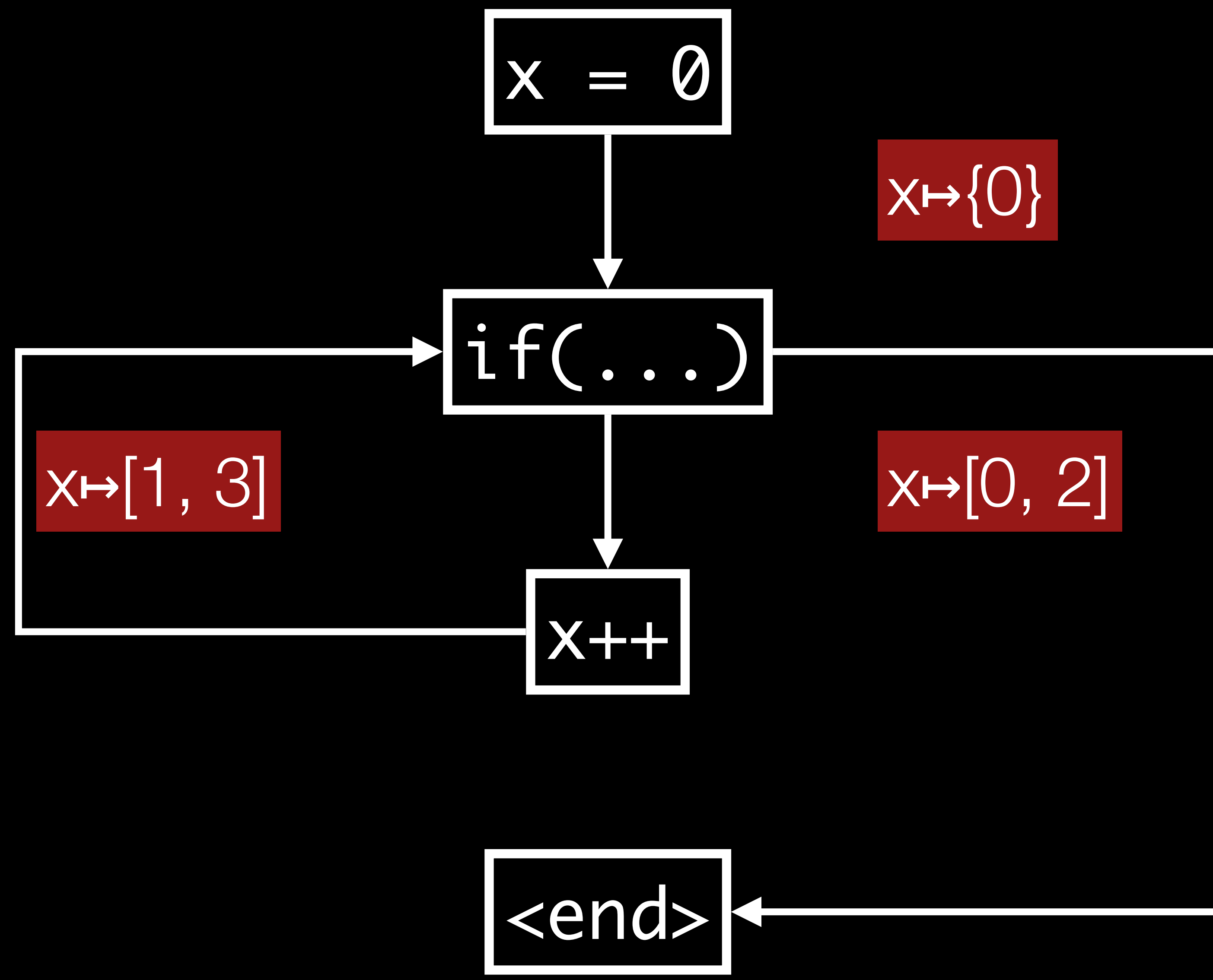
2. Analysis Abstraction
Lattice Example





2. Analysis Abstraction
Lattice Example





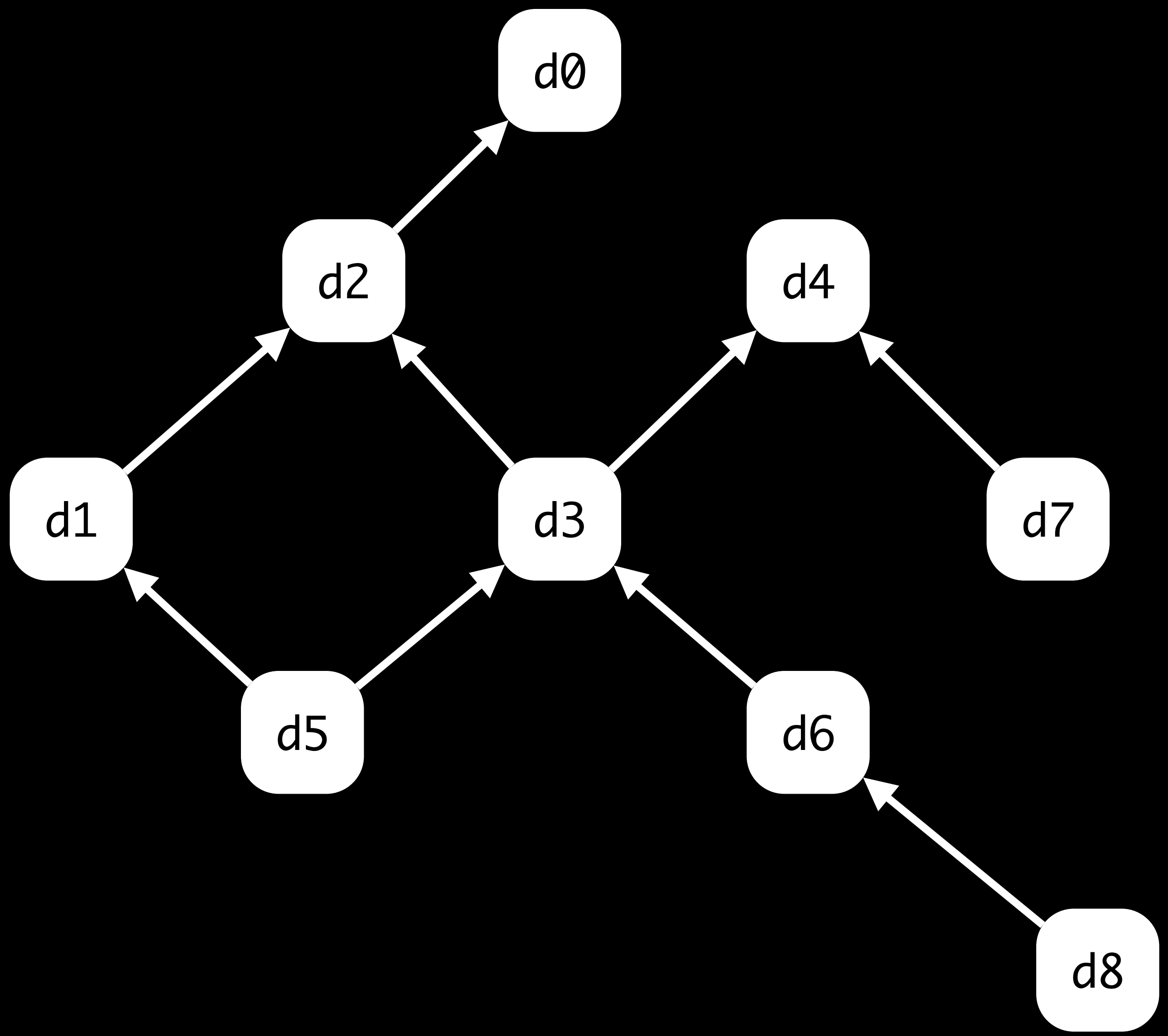
2. Analysis Abstraction
Lattice Hacks

- $x \sqsubseteq y$ if and only if $x \sqcup y = y$
- If $a \sqsubseteq b$ and $c \sqsubseteq d$ then $a \sqcup c \sqsubseteq b \sqcup d$
- $x \sqcup x = x$

3. Flow Functions

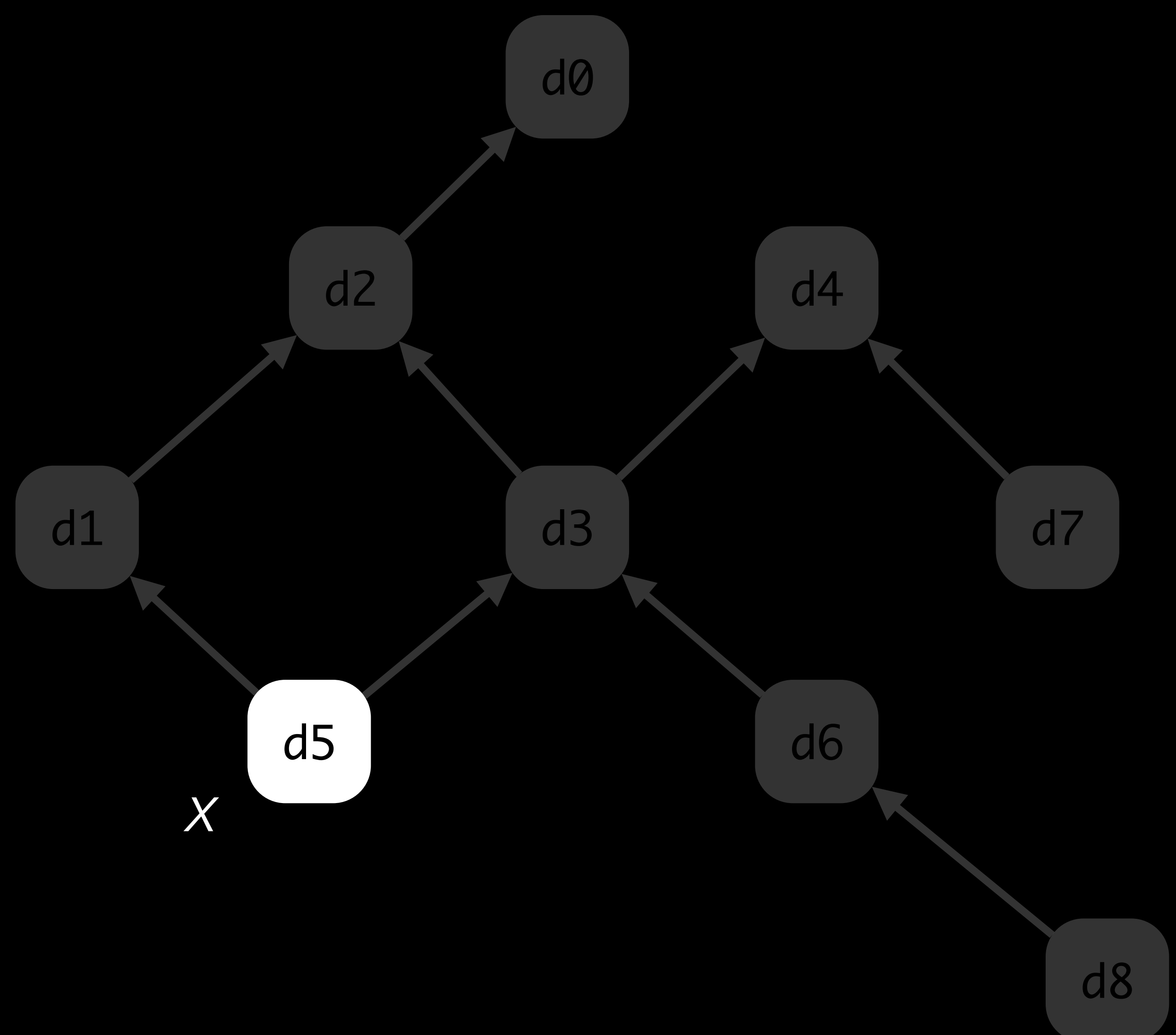
3. Flow Functions
Monotonicity

- If $(U, \sqsubseteq)$ is a lattice, then the function f is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \Longrightarrow f(x) \sqsubseteq f(y)$



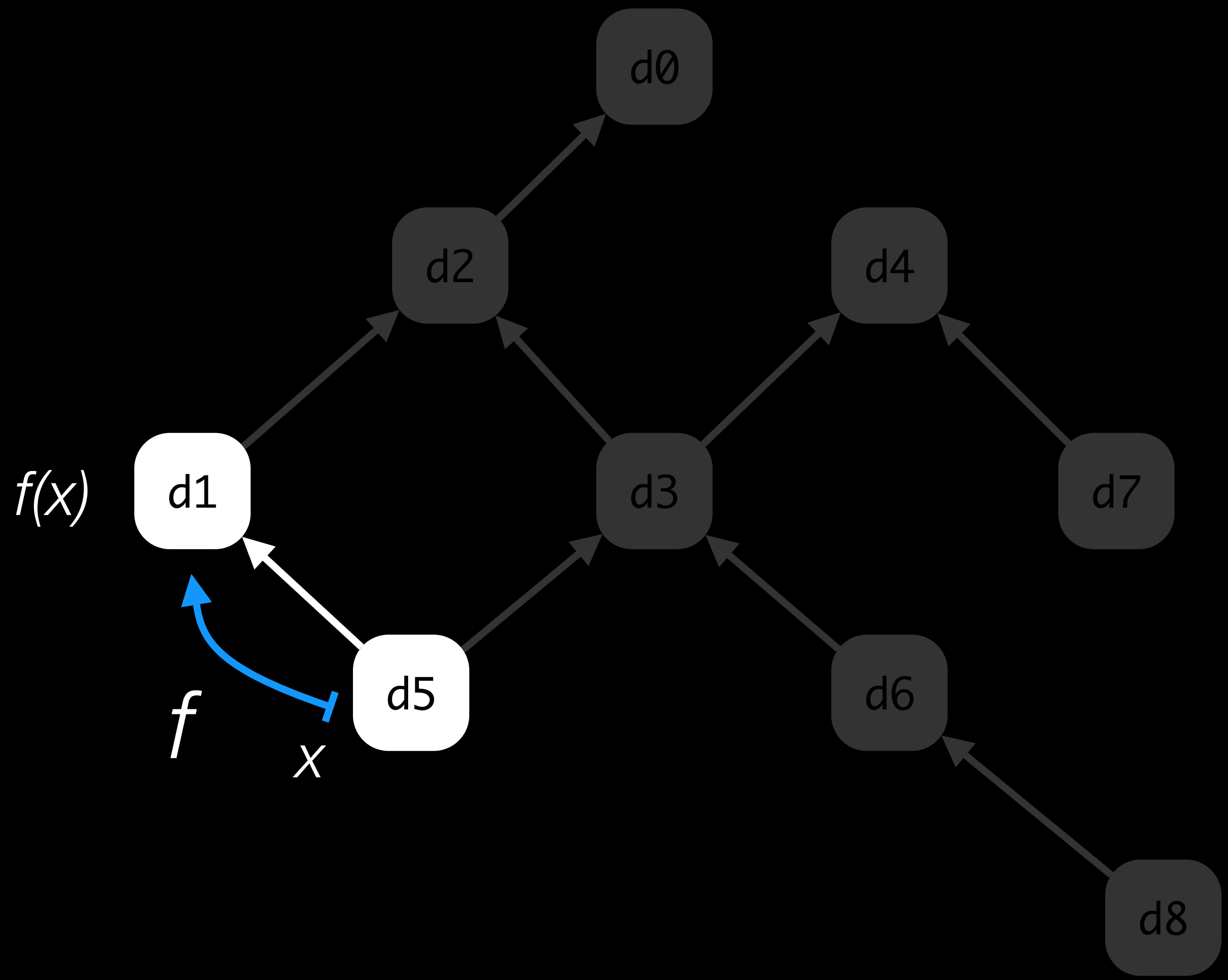
3. Flow Functions
Monotonicity

- If $(U, \sqsubseteq)$ is a lattice, then the function f is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \Longrightarrow f(x) \sqsubseteq f(y)$



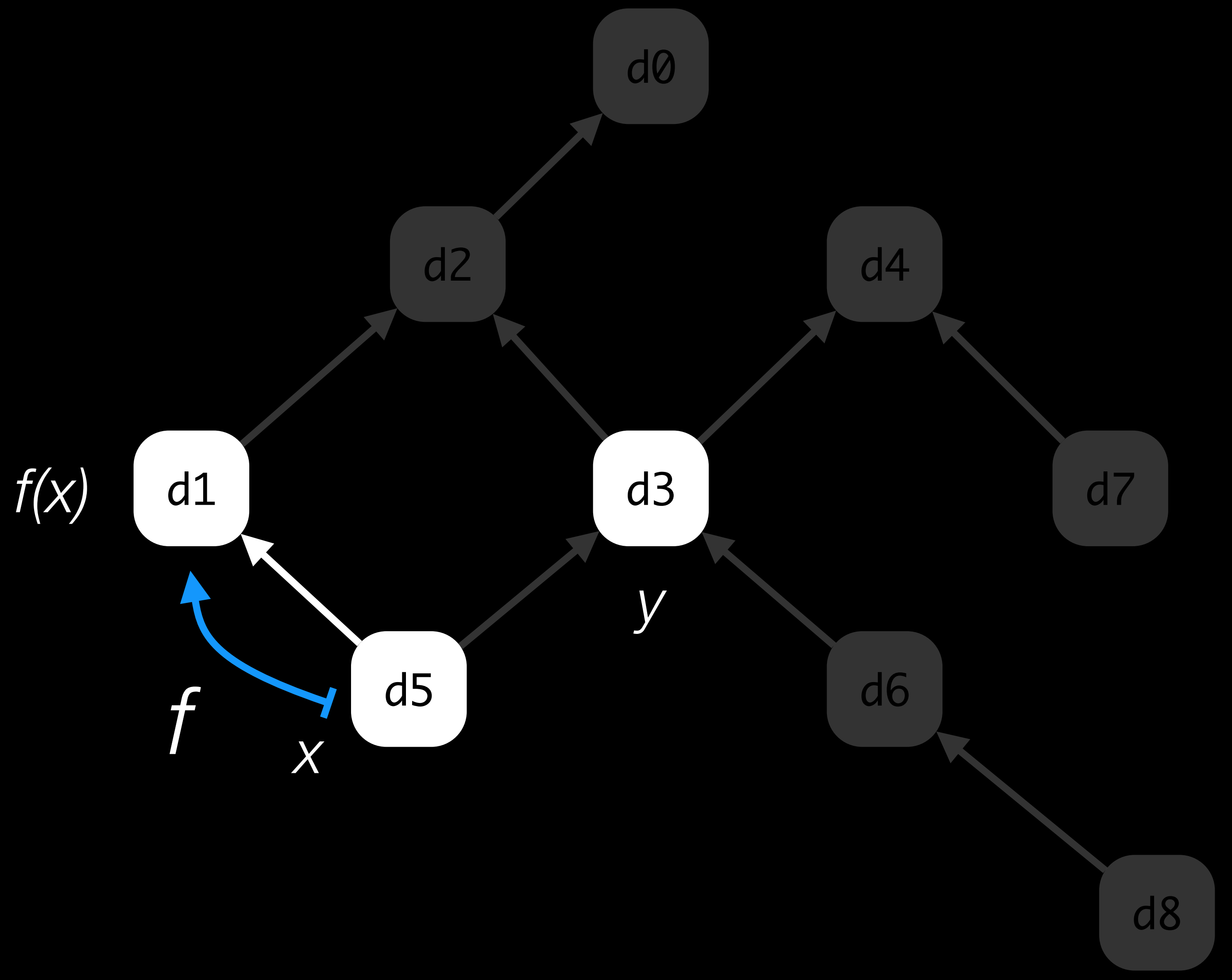
3. Flow Functions
Monotonicity

- If $(U, \sqsubseteq)$ is a lattice, then the function *f* is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$



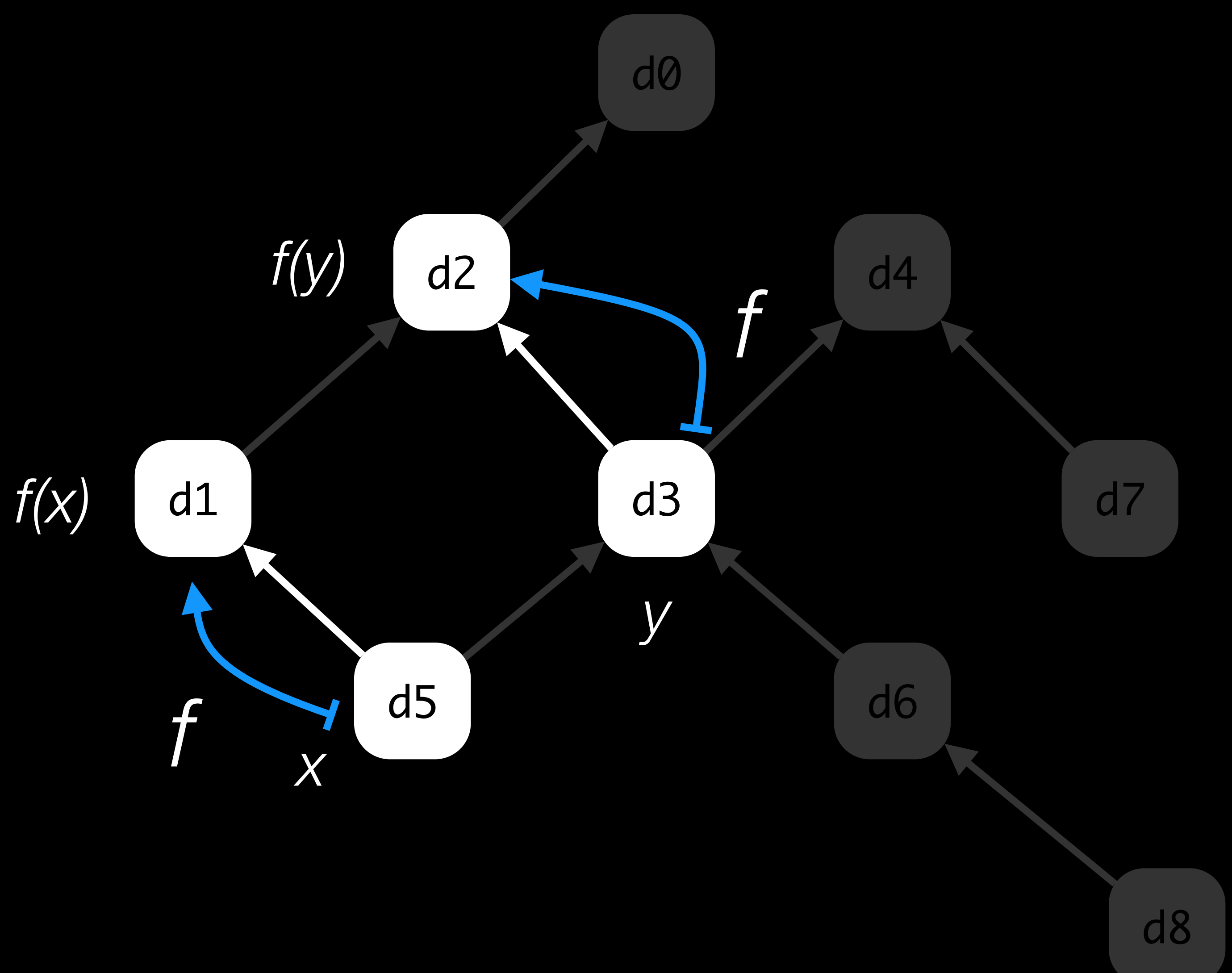
3. Flow Functions
Monotonicity

- If $(U, \sqsubseteq)$ is a lattice, then the function f is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$



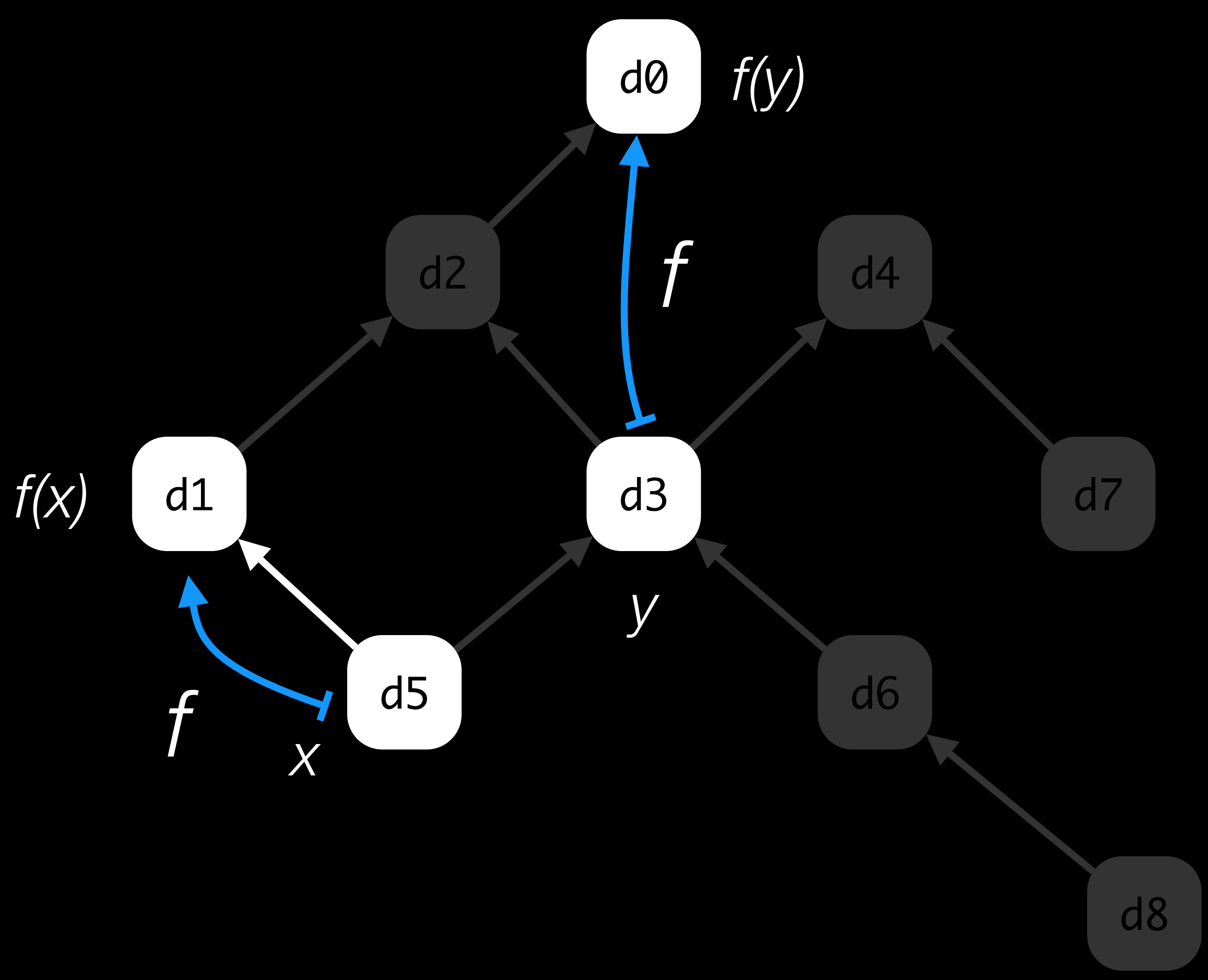
3. Flow Functions
Monotonicity

- If $(U, \sqsubseteq)$ is a lattice, then the function f is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$



3. Flow Functions
Monotonicity

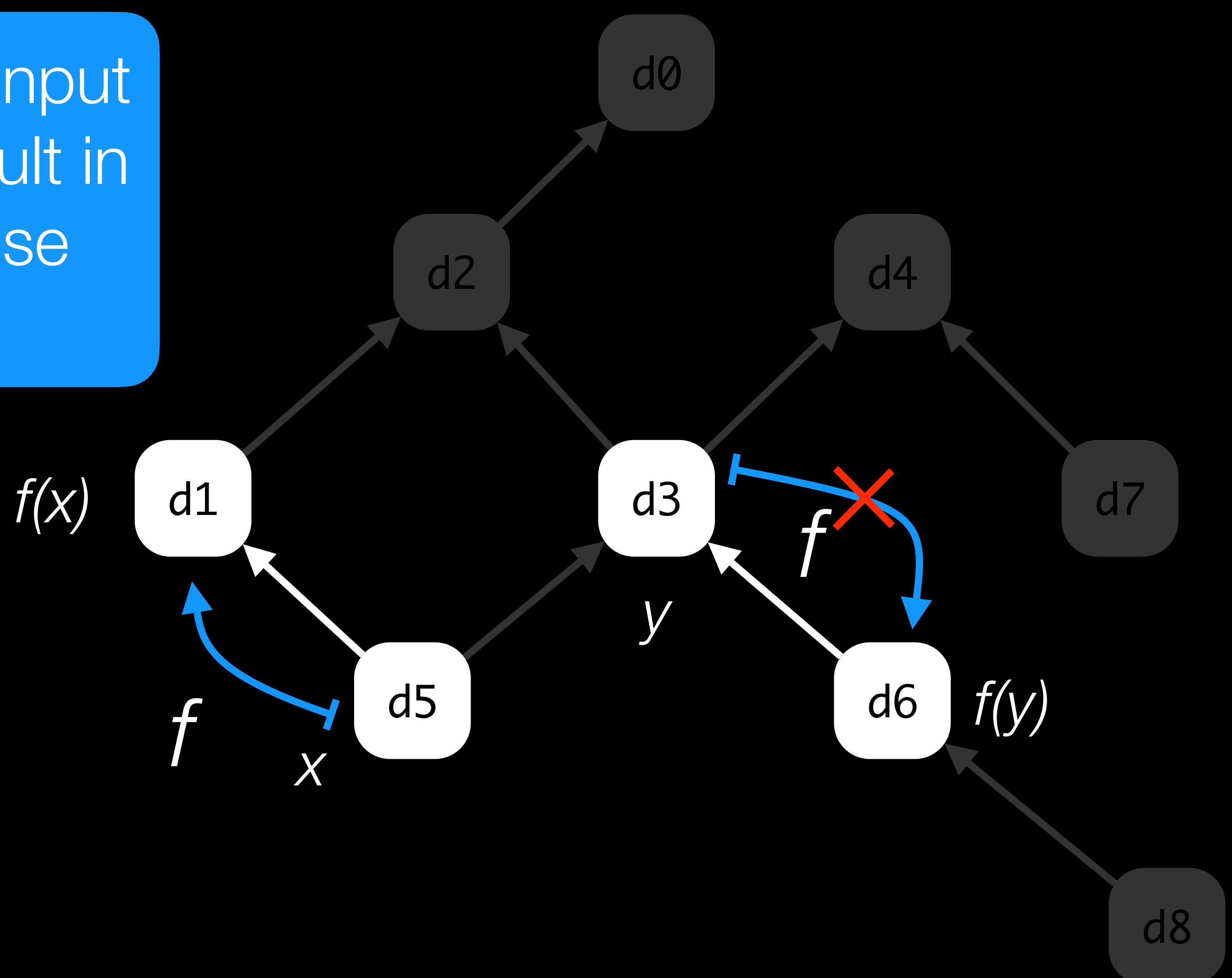
- If (U, $\sqsubseteq$) is a lattice, then the function f is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \Longrightarrow f(x) \sqsubseteq f(y)$



3. Flow Functions
Monotonicity

- If (U, $\sqsubseteq$) is a lattice, then the function *f* is monotone (i.e., order preserving) if:
  - $\forall\, x, y \in U : x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$

less precise input does not result in more precise output



putting it all together!

Lattice Fixed Point Theorem

Alfred Tarski
1955

Monotone Framework

- For each statement S in the control-flow graph, define a $f_S : L \to L$.
- For a path $P = S_0 S_1 S_2 \ldots S_n$ through the CFG, define $f_p(x) = f_n(\ldots f_2(f_1(f_0(x))))$.
- Goal: find the join-over-all-paths (MOP)

$$\mathrm{MOP}(n, x) = \bigsqcup$$

Generally Uncomputable
[Kam, Ullman 1977]

Monotone Framework

- For each statement S in the control-flow graph, define a $f_S : L \rightarrow L$.
- Goal: for each statement S in the CFG, find $V_{Sin} \in L$ and $V_{Sout} \in L$ satisfying

$$V_{Sout} = f_S(V_{Sin})$$

Least-Fixed-Point (LFP)

$$V_{Sin} = \bigsqcup_{P \in \text{Predecessors}(S)} V_{Pout}$$

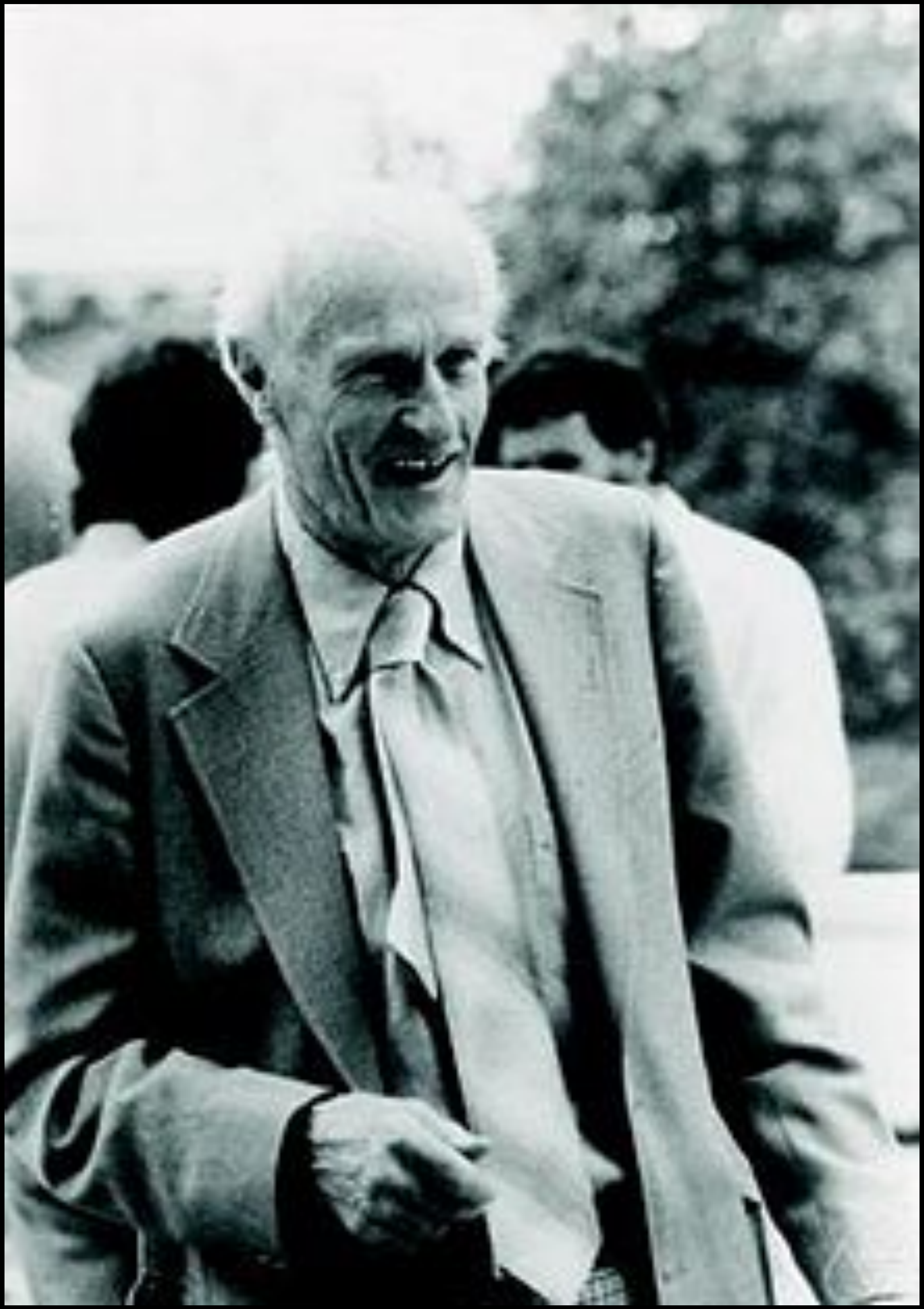
MOP(n, x) ⊑ LFP(n, x)

Generic Dataflow Algorithm

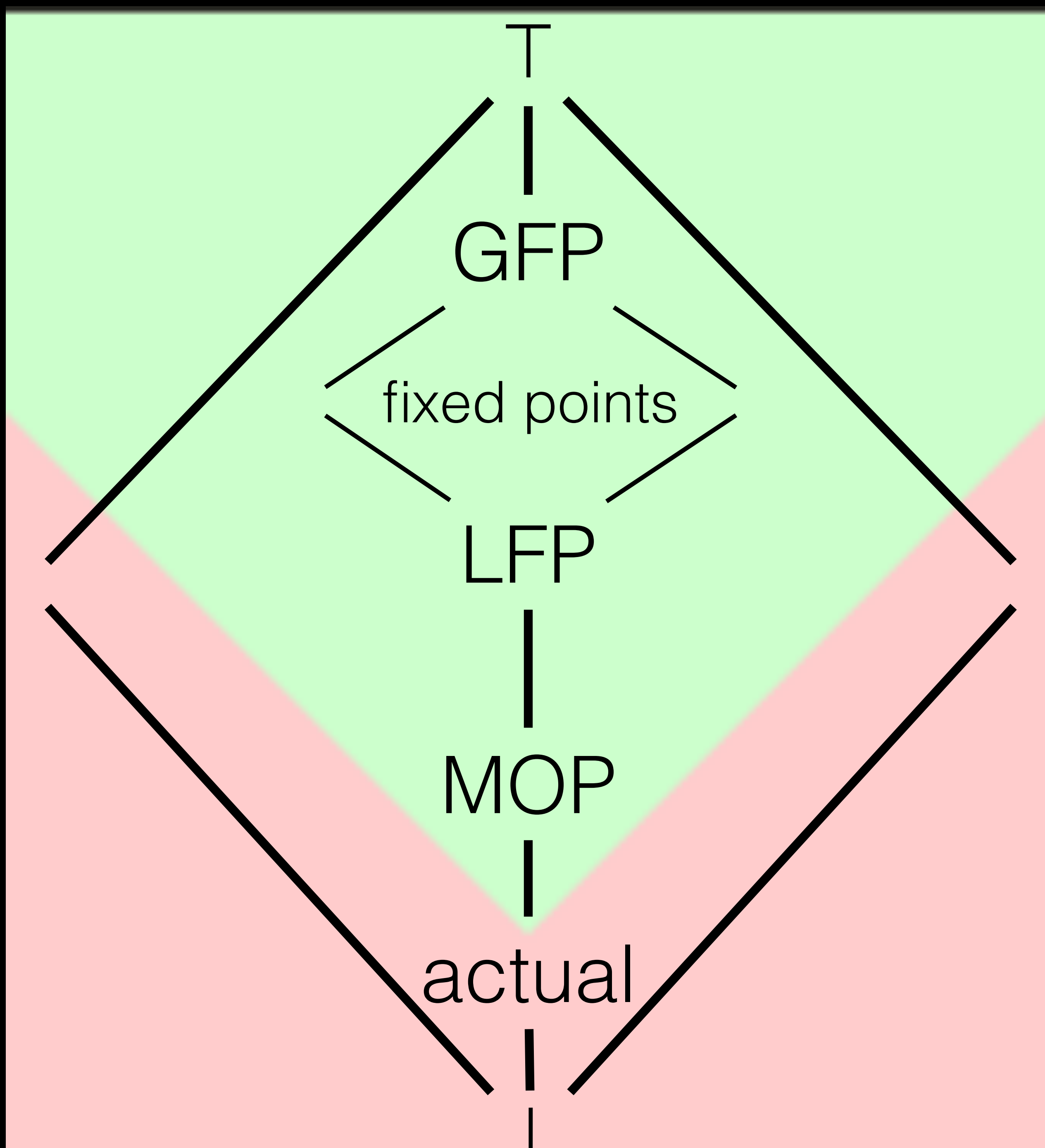
```
initialize out[s] = in[s] = ⊥ for all s
add all statements to worklist
while worklist not empty
  remove s from worklist
  in[s] = ⊔ p∈PRED(s) . out[p]
  out[s] = f_s(in[s])
  if out[s] has changed
    add successors of s to worklist
  end if
end while
```

Kleene Fixed Point Theorem

Stephen Cole Kleene
1938



MOP ⊑ LFP



- Every solution S ⊒ *actual* is “safe” (i.e., sound).
- MOP ⊒ *actual*
- LFP ⊒ MOP
- A flow function *f* is distributive if *f*(x) ⊔ *f*(y) = *f*(x ⊔ y)
- If all flow functions are distributive, then LFP = MOP
- Initializing using ⊤ instead of ⊥ causes earlier termination, but yields more imprecise fixed-point

Next

- Call Graph Construction